

АННОТАЦИЯ

62 с./ 66 с., 27 рис., 9 табл., 14 источников, 1 прил., 6 граф. матер.

В дипломном проекте разработана система позиционирования для мобильного робота, предоставляющая интерфейс управления и мониторинга состояния, а так же прошивку для коммутационной платы мобильного робота. Система предназначена для решения задачи позиционирования мобильного робота на основе относительных и абсолютных методов.

В первом разделе произведен обзор методов и алгоритмов позиционирования. Во втором разделе разработано техническое задание на разрабатываемую систему. В третьем разделе описана структура разрабатываемого программного обеспечения, а так же структура используемых в разработке библиотек. В четвёртом разделе описаны средства разработки, описано взаимодействие классов, приведена диаграмма классов, описаны алгоритмы функционирования модулей системы. В пятом разделе представлены результаты испытаний системы. Шестой раздел содержит расчет экономического эффекта от внедрения системы. В седьмом разделе представлен обзор современных тенденций по энергосбережению в области информационных технологий.

Титульный

Задание

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ.....	7
1.1 Система позиционирования с использованием датчиков оборотов колес	10
1.2 Инерционные системы позиционирования.....	10
1.3 Позиционирование на основе машинного зрения.....	12
1.4 Триангуляция по средствам точек беспроводного доступа.....	13
1.5 Сравнение моделей.....	14
1.6 Подход на основе Data fusion.....	14
2 РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ.....	16
2.1 Требования к системе в целом.....	16
2.2 Требование к задачам (функциям) программных средств.....	17
2.3 Требования к видам обеспечения.....	18
2.4 Перечень документов на разработку.....	19
3 СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.....	21
3.1 Организация системы.....	21
3.2 Описание модулей общей системы.....	21
3.3 Обзор библиотеки Qt.....	23
3.4 Структура Qt и использованные классы.....	25
3.5 Обзор библиотеки OpenCV.....	27
3.6 Обзор платформы для управления аппаратным обеспечением Arduino.....	29
4 ОПИСАНИЕ РЕАЛИЗАЦИИ ПРОГРАММНОЙ СИСТЕМЫ И АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ.....	31
4.1 Описание среды разработки Qt Creator.....	31
4.2 Выделение классов и взаимосвязей между ними.....	32
4.3 Описание использованных алгоритмов.....	33
4.4 Описание реализованных классов.....	36
5 РЕЗУЛЬТАТЫ ИСПЫТАНИЙ СИСТЕМЫ.....	38
6 РАСЧЕТ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ.....	45
6.1 Расчет общей трудоемкости разработки программного обеспечения.....	45
6.2 Расчет заработной платы разработчиков программного обеспечения.....	48
6.3 Расчет себестоимости и отпускной цены программного обеспечения.....	50
7 ЭНЕРГО И РЕСУРСОСБЕРЕЖЕНИЕ.....	57

					<i>ДП.АС24.06791 – 11 81 00</i>			
Изм	Лист	№ докум.	Подп.	Дата				
Разраб.		Дёмин В. В.			РАЗРАБОТКА ПРОГРАММНЫХ СРЕДСТВ ПОЗИЦИОНИРОВАНИЯ МОБИЛЬНОГО РОБОТА. Пояснительная записка			
Проверил		Дунец А. П.						
И контр.		Крапивин Ю. Б.						
Утв.		Головкин В. А.						
						<i>Лит</i>	<i>Лист</i>	<i>Листов</i>
						Д	4	62
						УО «БрГТУ»		

ЗАКЛЮЧЕНИЕ.....	60
СПИСОК СОКРАЩЕНИЙ.....	61
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	62
ПРИЛОЖЕНИЕ А — ТЕКСТ ПРОГРАММЫ.....	63

					ДП.АС24.06791 – 11 81 00	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

ВВЕДЕНИЕ

Современный уровень развития вычислительной техники превращает в реальность давнюю мечту человечества — полностью автоматизированные промышленные линии, выросшие из простейших механизмов. Первоначальные мечты об освобождении человека от физического труда, были окончательно решены с появлением индивидуального электропривода, позволив механизировать не только энергетику станков, но и управление ими. На этой основе возникли и получили развитие разнообразные станки-автоматы, а затем и первые простейшие автоматические линии. Следующим этапом автоматизации промышленности стало появление микропроцессоров, позволяющих снять с человека функцию непосредственного управления оборудованием, заменив ее на функцию контроля. Но современный уровень развития вычислительной техники позволяет так же, автоматизировать функции контроля за работой системы и качеством произведенного продукта. Для этих целей автоматические линии оборудуются различными сенсорами, позволяющие оценить тот или иной параметр для проверки результата работы.

В связи с развитием робототехники в автоматизированном производстве внедряется все большее количество мобильных роботизированных платформ, выполняющих самые разнообразные действия: от перевозки грузов с одного конвейера на другой, до применения в качестве автоматизированного шасси укомплектованного различным оборудованием. Можно предположить, что со временем мобильные платформы полностью заменят конвейерные ленты, за счет своей высокой мобильности и легкой модификации процессов производства.

Одной из самых сложных и основополагающих задач, не имеющих универсального решения является задача навигации мобильного робота. Она включает в себя проектирование системы датчиков и программного обеспечения. Каждая система разрабатывается исходя из поставленной задачи, поэтому не существует стандартной архитектуры и её необходимо разрабатывать самостоятельно [1]. Большинство систем позиционирования, имеющих практическое применение имеют ряд препятствий на пути к их изучению и пониманию: закрытые исходные коды системы, дорогие датчики и оборудование, проприетарные средства для прошивки, отладки и симуляции системы. Такие трудности делают разработку системы позиционирования актуальной и необходимой при разработке мобильного робота.

1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ

Основная задача позиционирования — имея данные об окружающем пространстве (карта местности, маршрут передвижения относительно текущего положения, ориентиры на местности) обеспечить необходимую точность передвижения для достижения конечной точки маршрута.

Современные системы позиционирования можно разделить на два вида [2]:

- 1) относительные, позиция в которых определяется относительно начального положения объекта;
- 2) абсолютные, позиция в которых определяется на основании данных окружающей среды.

Системы позиционирования мобильных роботов

Относительные системы позиционирования требуют минимальное количество датчиков для построения пройденного пути. Такой вид позиционирования является достаточно точным, при условии, что робот будет сверять своё положение на контрольных точках.

Относительные системы позиционирования представлены двумя категориями:

- 1) одометры (датчики оборотов колёс) (см. рисунок 1.1);
- 2) инерционное позиционирование (акселерометры (см. рисунок 1.2), гироскопы, датчики ускорения, датчики угла поворота, компасы).



Рисунок 1.1 — Одометр правого колеса мобильного робота

Другим видом позиционирования является абсолютное позиционирование. Оно представлено следующими категориями:

- 1) Активные маяки, триангуляция (радиомаяки, инфракрасные (ИК) маяки, Wireless Fidelity (Wi-Fi) маяки (см. рисунок 1.3), Global System for Mobile Communications (GSM) маяки, Global Positioning System (GPS)).



Рисунок 1.2 — Акселерометр измеряющий ускорения в трех измерениях



Рисунок 1.3 — Определение местоположения по GSM маякам

- 2) Искусственные паттерны (метки, маркеры) для распознавания (см. рисунок 1.4).
- 3) Естественные паттерны (маркеры) для распознавания (см. рисунок 1.5).

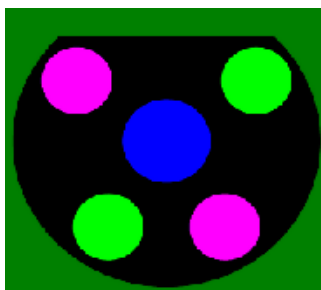


Рисунок 1.4 — Искусственный паттерн соревнований Robocup Soccer Small League



Рисунок 1.5 — Объект «водонапорная башня» используется как уникальный естественный паттерн на когнитивной карте местности

- 4) Сравнение моделей или Simultaneous Localization and Mapping (SLAM) алгоритмы.
- 5) Подход на основе Data fusion (см. рисунок 1.6).

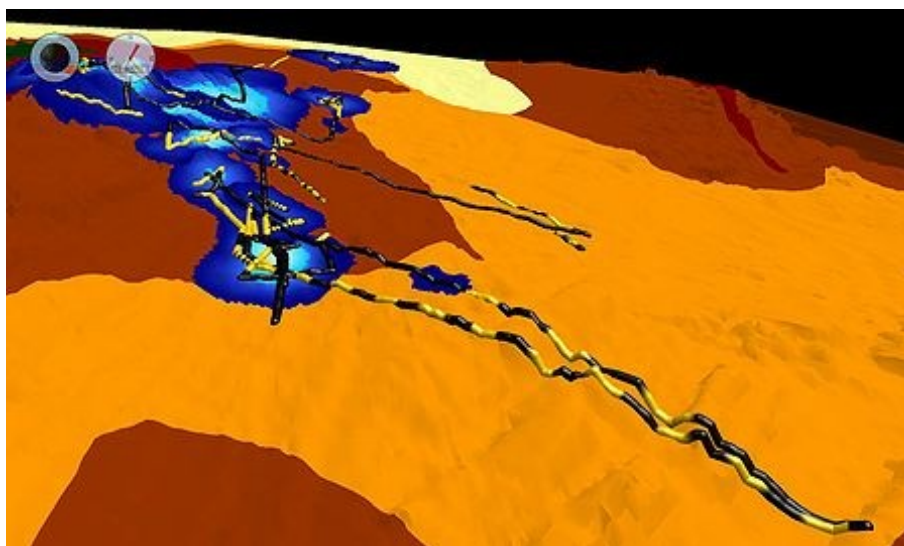


Рисунок 1.6 — Когнитивная карта местности построенная на основе подхода Data Fusion

1.1 Система позиционирования с использованием датчиков оборотов колес

Современные серводвигатели, которыми комплектуют промышленных роботов, имеют встроенную возможность определять количество оборотов колеса (одометры), тем самым предоставляя возможность анализировать пройденное расстояние [3]. Несмотря на то что получаемые результаты имеют высокую точность, как показатель перемещения робота они постепенно накапливают погрешности. Неровная поверхность пола, вибрация его поверхности, столкновения с препятствиями вносят незначительные погрешности в измерения, и в исключительных случаях (при возникновении коллизий) не могут отслеживать эти ситуации. Поэтому одометры должны использоваться в совокупности с другими датчиками, позволяя отслеживать происходившие коллизии на пути передвижения робота и вносить поправки в расчеты маршрута, тем самым, компенсируя накапливающиеся погрешности.

1.2 Инерционные системы позиционирования

Основой систем инерционного позиционирования служат акселерометр и гироскоп.

Акселерометр — прибор, измеряющий проекцию кажущегося ускорения. Кажущееся ускорение есть ускорение, вызванное равнодействующей сил не гравитационной природы, действующая на массу и равное этой силе отнесённой к величине этой массы. Современные акселерометры позволяют измерять ускорение сразу в трех плоскостях.

Гироскоп — устройство, способное измерять изменение углов ориентации связанного с ним тела относительно инерциальной системы координат, как правило основанное на законе сохранения вращательного момента (момента импульса).

Гироскоп чаще всего применяется как чувствительный элемент указывающих гироскопических приборов и как датчик угла поворота или угловой скорости для устройств автоматического управления. В некоторых случаях, например в гиростабилизаторах, они используются как генераторы момента силы или энергии.

Гироскопы имеют незначительную погрешность. Поэтому определение направленности робота во время движения становится решаемой проблемой, даже на основе недорогих датчиков. Акселерометр же имеет свойство накапливать большие погрешности, тем самым увеличивая радиус возможного положения робота и не подходит в виде основного датчика, а только в виде вспомогательного. Большая погрешность возникает из-за необходимости двойного интегрирования получаемых

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

данных ускорения для получения координат. Тем самым накапливается квадратичная погрешность.

Основные области применения гироскопов — судоходство, авиация, робототехника и космонавтика.

Инерциальная навигация относится к такому способу определения местоположения в пространстве, при котором не используются данные каких-либо внешних источников. Все чувствительные элементы находятся непосредственно на борту транспортного средства. Инерциальные измерители линейных ускорений - акселерометры установлены на так называемой гиростабилизированной платформе. Эта платформа, используя свойства гироскопа - сохранять неизменной ориентацию своей оси в пространстве, обеспечивает строго горизонтальное положение осей чувствительности акселерометров (с точностью до единиц угловых секунд). Измеренные ускорения дважды интегрируются, и, таким образом, вычисляется приращение положения объекта. Объединенные общей задачей определения координат подвижного объекта, гироскопы и акселерометры образуют инерциальную навигационную систему (ИНС). Помимо этой задачи ИНС предоставляет информацию об угловой ориентации объекта: углах крена, тангажа и рыскания (курса), и о скорости объекта [4].

Конструкция современной ИНС вобрала в себя последние достижения точной механики, теории автоматического управления, электроники и вычислительной техники.

Конструктивно ИНС можно разделить на два класса: платформенные и бескарданные. В первых гиростабилизированная платформа реализована физически в виде рамы трехстепенного карданного подвеса. В таких системах используются традиционные гироскопы с вращающимся ротором. Точность таких систем может достигать 1 морской мили (900 м) за час работы. Эти системы входят в состав бортового навигационного оборудования тяжелых самолетов.

Другой класс - бесплатформенные ИНС (БИНС) отличаются тем, что плоскость горизонта в них реализована математически, используя данные гироскопов и акселерометров. В этих системах могут быть использованы лазерные и волоконно-оптические гироскопы. Здесь нет вращающихся частей, а об угловой скорости судят по фазовой задержке лазерного луча пробегающего по замкнутому контуру. Точность этих систем 1 морская миля за час. Они существенно конструктивно проще и дешевле платформенных. Лучшие образцы БИНС способны показывать точность, сравнимую с точностью платформенных систем.

Но как и любая система позиционирования, ИНС имеет свои недостатки. Решение задачи ИНС, которая уберёт эти недостатки может быть достигнуто как при помощи добавления новых датчиков, так и созданием архитектуры более совершенного программного обеспечения.

1.3 Позиционирование на основе машинного зрения

На сегодняшний день существует множество реализаций машинного зрения, но ни одна из них не является универсальной [5]. На их основе мобильные-роботы могут перемещаться в пространстве избегая препятствий, распознавать располагающиеся в пространстве объекты, а так же строить карту окружающего мира. Ежегодно проводятся соревнования американской военной компанией Defense Advanced Research Projects Agency (DARPA), соревнования Robocup в которых используются последние наработки машинного зрения для решения поставленных задач. Разработки, имеющие большую эффективность имеют своим продолжением реализацию в промышленных, военных и медицинских роботах и являются эффективными системами позиционирования.

В подходе на основе машинного зрения выделяют два подхода — на основе естественных и на основе искусственных паттернов. При подходе с использованием искусственных паттернов на местности располагаются известные роботу отличительные метки для распознавания, каждая из которых имеет свои координаты. Для определения текущего положения необходимо что бы в видимости находились 3 и более паттерна. Несмотря на то что погрешности в определении положения невелики, не всегда в поле зрения находится необходимое количество меток. Погрешность может возникать из за ошибок в оценке геометрических размеров маркеров. Так же достаточно распространены радио метки и штрих-коды. Например, их используют такие метод как Radio Frequency Identification (RFID) (радиочастотная идентификация) — метод автоматической идентификации объектов, в котором по средствам радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID метках; одним из успешных применений искусственных меток на практике роботизированный склад разработанный компанией Kiva Systems, где паттерном является метка с QR-кодом (Quick Response) (рис 1.7).

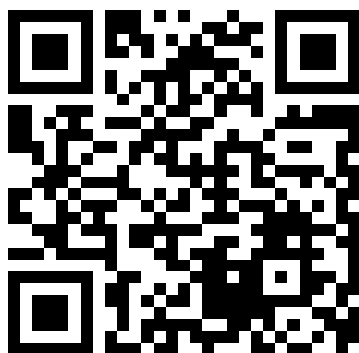


Рисунок 1.7 — Метка с QR-кодом

QR-код — матричный код (двухмерный штрихкод), разработанный и представленный японской компанией Denso-Wave.

Объем информации, зашифрованной в коде перестал устраивать индустрию. Были поставлены эксперименты с новыми способами кодирования небольших объемов информации в графической картинке — в результате был разработан QR-код. В начале 2000 года получил широкое распространение в Японии. Его можно было встретить на большом количестве плакатов, упаковок и товаров. Основное достоинство QR-кода — это легкое распознавание сканирующим оборудованием (в том числе и фотокамерой мобильного телефона), что дает возможность использования в торговле, производстве, логистике, а так же робототехнике.

Максимальное количество символов, которые помещаются в один QR-код:

- Цифры — 7089;
- Цифры и буквы (включая кириллицу) — 4296;
- Двоичный код — 2953 байт;
- Иероглифы — 1817.

Таким образом робот в автоматизированном складе от Kiva Systems может получить исчерпывающую информацию об захватываемом стеллаже.

В подходе с естественными метками выступают отличительные особенности окружающей среды. Для робота нет необходимости специально готовить местность - наносить маркеры, но требуется иметь представление о местности (когнитивную карту). Как правило главным ориентиром на местности выступают большие объекты, к примеру водонапорная башня изображенная на рисунке 1.5. Надежность этого метода намного ниже чем метода с искусственными паттернами.

1.4 Триангуляция по средствам точек беспроводного доступа

В области робототехники локация по средствам триангуляции от трех точек беспроводного доступа получила широкое развитие, т.к. точность её достаточно высока, а затраты сводятся к размещению маяков и задание их точных координат. Можно найти множество научных исследований и инженерных решений, основанных на контроллерах ZigBee, Wi-Fi и радиоточках. Триангуляция относится к абсолютному позиционированию, поэтому её использование может всегда дать координаты с минимальной погрешностью и точное положение относительно имеющейся карты местности. Тем самым давая возможность роботу на основе других датчиков корректировать свои знания относительно своего местоположения.

1.5 Сравнение моделей

Объект изучает окружающую среду при помощи имеющихся датчиков и на основе полученных данных составляет карту мира. Далее объект сравнивает сформированную карту местности с имеющейся и на этой основе определяет своё местоположение. В основе лежит два вида карт местности: геометрические и топологические. В геометрических картах мир представлен как глобальная система координат, когда в топологических как сеть узлов и дуг — графом. Одним из таких подходов являются Simultaneous localization and mapping (SLAM) алгоритмы — позволяют в неизвестном пространстве определять стабильные метки, строить карту пространства и одновременно определять свои положения на этой карте. Простейший SLAM алгоритм использует фильтр Калмана [6], определяя на основе увиденных координат меток свои координаты. Алгоритм как правило самообучается, тем самым постепенно адаптируясь к окружающей среде.

SLAM алгоритмы состоят из нескольких частей: поиска метки, обработки данных, оценки состояния, обновления состояния и обновления метки. Существует множество способов решить каждую из подзадач. Как следствие при реализации такого вида алгоритмов имеется несколько методов решающих одну и ту же подзадачу разными способами. Каждое решение используется при определенных условиях, в которых оно показывает наилучшие результаты по сравнению с другими. Тем самым SLAM алгоритмы адаптивно подстраиваются под условия окружающей среды.

Важным аспектом при реализации такого вида алгоритмов является аппаратное обеспечение робота, т. к. алгоритмы навигации колесного робота и алгоритмы навигации гуманоидного робота значительно отличаются. Основным критерием для такой системы является одометрическая производительность измерений — набор датчиков используемых для определения координат робота и траектории его перемещения.

1.6 Подход на основе Data fusion

Информация предоставляемая одним сенсором обычно ограничена и может иметь большие погрешности. Использование нескольких сенсоров является способом для повышения точности, а так же возможностью получения дополнительной информации о среде в которой находится робот. Такой подход получил название Data Fusion.

Концепция подхода заключается в объединение данных в одну модель, создавая из множества однотипных данных о разных воздействиях на сенсоры, общую модель. При объединении, с однотипных данных происходит нормализация, приведение к

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

одному измерению. Это позволяет сравнить полученные результаты с нескольких датчиков, и определить возможные погрешности в измерениях, а так же исправность самих датчиков. Для неоднородных данных используют различные подходы для анализа и совмещения данных на основе фильтров Калмана, нечеткой логики, мультиагентного подхода. Тем самым из множества простых составляющих получается сложная модель, свойств у которой больше чем сумма исходных свойств совмещённых данных. Возможности позиционирования на основе данной модели значительно расширяются. Такой подход позволяет создавать на основе бюджетных датчиков точные системы позиционирования, что сокращает стоимость системы в целом и повышает ее надежность и доступность конечному пользователю.

Фильтр Калмана это эффективный рекурсивный фильтр, который оценивает состояние линейной динамической системы по серии неточных измерений. Он используется в широком спектре задач от радаров до систем технического зрения, и является важной частью теории управления системами.

Фильтр Калмана является разновидностью рекурсивного фильтра. Это означает, что только результат предыдущей итерации фильтра (в виде оценки состояния системы и оценки погрешности определения этого состояния) и текущие наблюдения нужны для расчета текущего состояния системы. В отличие от пакетных фильтров не требуется хранение никакой истории наблюдений.

Состояние фильтра содержится в двух переменных:

- $\mathbf{x}_{k|k}$, оценочное состояние системы в момент времени k , которое получено на основании начального состояния системы и всех наблюдений по момент времени k включительно;
- $\mathbf{P}_{k|k}$, матрица ковариаций этого состояния, включающая в себя оценку дисперсий погрешности вычисленного состояния и уровни ковариаций, показывающих выявленные взаимосвязи между параметрами состояния системы.

Итерация фильтра Калмана делится на две фазы: предсказание и учет наблюдений. Фаза предсказания использует вычисленное на предыдущем шаге состояние для получения через модель системы оценочного состояния на текущем шаге. В фазе учета наблюдения информация об измерениях произведенных на текущем шаге используется для уточнения информации о состоянии системы, что делает её в результате более точной.

Сложность применения модели Data Fusion заключается в затратах на анализ, применение методов совмещения и калибровки подготовленной системы. Эти процессы являются наукоемкими и требуют квалифицированных разработчиков для проектирования и реализации такой системы. Эффективность применения такого подхода заключается в повышении точности измерения уже существующей системы.

2 РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

2.1 Требования к системе в целом

Цель создания системы — автоматизация процесса навигации робота в пространстве; создание программных средств навигации мобильного робота на основе видеокамеры и одометров управляющей им как по заданию пользователя, так и в автономном режиме, которая смогла бы детектировать робота в помещении, определять его координаты, управлять передвижением робота в помещении, строить график пройденного пути по совмещенным данным камеры и одометров; обеспечение надежного способа детектирования робота; возможность интеграции разработанных программных средств в другие программные средства робототехники; переносимость между различными платформами.

1) Требования к структуре программных средств:

а) Техническая архитектура программных средств должна быть представлена в виде персональной электронной вычислительной машины (ПЭВМ), состоящей из компьютера, Bluetooth устройства передачи данных, цифровой видеокамеры и мобильного робота на основе мобильной колесной платформы с одометрами ведущих колес, коммутационной платой Arduino и беспроводной связью Bluetooth.

б) Программная архитектура должна быть представлена в виде программных средств управления (управляющая система — объект управления). Системное программное обеспечение (ПО) будет включать в себя многопользовательскую многозадачную многопроцессорную систему разделения времени операционную систему (ОС) GNU (GNU's Not UNIX) Debian Linux Wheezy, инструментальное ПО представлено прошивкой Arduino.

в) Информационная архитектура программных средств должна быть представлена внемашинной информационной архитектурой и внутримашинной информационной архитектурой. Внутримашинная информационная архитектура должна быть представлена в виде структуры хранения настроек программных средств (матрица искажений цифровой видеокамеры, значения диапазонов цветов детектирования). Внемашинная информационная архитектура должна быть представлена внемашинной информационной структурой (структура хранения информации пройденного пути).

г) Организационная архитектура должна быть представлена пользователями (инженер по настройке, оператор) и специалистами по эксплуатации и сопровождению программных средств (инженер-программист).

д) Внешняя среда должна быть представлена конечными пользователями (инженер программист, инженер-системотехник).

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

2) Требования к численности и квалификации персонала программных средств позиционирования мобильного робота.

Для успешного эксплуатации программных средств необходим как минимум один инженер-программист с высшим образованием, имеющий опыт работы с мобильными роботами и программными средствами их управления, а также инженер-системотехник – обслуживание технического обеспечения.

К типам пользователей относятся следующие:

Инженер по настройке — производит калибровку и настройку программных средств позиционирования мобильного робота под условия окружающей среды, производит инициализацию робота и выбор паттерна для распознавания;

Оператор — с помощью программных средств задает контрольные точки роботу перемещения по помещению, производит мониторинг работы, строит и анализирует графики пройденного пути робота.

3) Требования к эксплуатации, техническому обслуживанию.

Запуск и остановка программных средств должны выполняться операторами. Обслуживание технических элементов (профилактика, ремонт, хранение), обслуживание и восстановление программных элементов, восстановление внутренних информационных элементов в случае сбоя должно выполняться инженером-программистом и инженером-системотехником.

5) Требования по сохранности информации при авариях.

Программные средства должны обеспечивать сохранность информации при авариях.

6) Требования по стандартизации и унификации.

Использование разработанного протокола передачи управляющих сигналов.

2.2 Требование к задачам (функциям) программных средств

1) Перечень функций, задач или их комплексов, подлежащих автоматизации по каждой подсистеме:

а) Модуль «Навигация»: организация детектирования робота на изображении с видеопотока, определение координат робота, реализация обратной связи с роботом.

б) Модуль «Управление мобильным роботом»: организация интерфейса для управления роботом по беспроводной связи Bluetooth, получения данных с одометров, инициализация.

в) Модуль «Построение графика пройденного пути» — настройка отображаемых результатов пройденного пути мобильным роботом.

г) Модуль «Мобильный робот» — управление периферией робота, беспроводная связь с модулем «Управление мобильным роботом».

2) Требования к выходной информации.

Выходные данные должны отображаться на экран монитора. Информация о текущих координатах робота должна отображаться в окне графического интерфейса. Графическое окно построения пройденного пути отображает траекторию пути робота.

2.3 Требования к видам обеспечения

1) Требования для информационного обеспечения программных средств:

Программные средства должны обеспечивать:

- а) Детектирование заданного паттерна системой, определение его текущих координат.
- б) Сохранение и восстановление настроек и данных калибровки камеры, диапазонов цветов паттернов, данных пройденного пути.
- в) Управление мобильным роботом.
- г) Построение пройденного пути робота в графическом виде.

2) Требования для программного обеспечения.

а) Программная архитектура должна быть построена на модульном принципе, взаимодействие между ними будет происходить на модульном уровне. Должен быть выделен основной модуль, в который будут интегрированы остальные. Главный модуль управляет роботом (объектом управления) через беспроводную связь Bluetooth.

б) Системное программное обеспечение будет включать в себя многопользовательскую многозадачную многопроцессорную сетевую систему разделения времени ОС GNU Debian Linux Wheezy, инструментальное программное обеспечение представлено прошивкой Arduino.

в) Программное обеспечение не должно зависеть от используемых средств вычислительной техники, систем коммуникационной техники и средств организационной техники. ПО должно быть кроссплатформенным компилироваться и собираться для ОС Windows, ОС Linux.

г) Программное обеспечение будет разрабатываться на языке высокого уровня C++ для компилятора GNU Compiler Collection (GCC) используемого в составе интегрированной среды разработки Qt Creator 2.1. Для разработки прошивки коммутационной управляющей платы робота будет использоваться Arduino IDE. Для написания прошивки используется язык высокого уровня — Processing.

д) Для взаимодействия пользователей с ПО используется графический интерфейс созданный на основе компонентов Qt 4 Framework, инструментом для

реализации которого является встроенный в Qt Creator дизайнер компонентов графического окна — Qt Designer.

е) Для взаимодействия с мобильным роботом будут использоваться следующие протоколы: RS232. Описание команд управления робота и способ их передачи будет подробно рассмотрен в разделе 3.

3) Требования для технического обеспечения программных средств:

Требования к техническому обеспечению представлены в таблице 2.1.

Таблица 2.1 — Требования к техническому обеспечению

Тип оборудования	Предполагаемое применение в программных средствах	Краткая теоретическая характеристика	Ориентировочная стоимость	Наименование предприятия-поставщика
Мобильный робот	Объект управления	Коммутационная плата Arduino, одометры, мобильная четырехколесная платформа	500 тыс. рублей	-
Цифровая видеокамера	Получение видеоизображения	Вебкамер Philips SPC630N	180 тыс. рублей	-
Персональный компьютер	Рабочее место инженера-программиста	Intel Pentium 4, HDD 300 Гб, ОЗУ 4096 Мб, Video 512 Мб, монитор TFT 19", 1280x1024, клавиатура Chicony KWD-205, PS/2 мышь A4Tech Optical Comfort, адаптер Bluetooth	4,1 млн. рублей	-

2.4 Перечень документов на разработку

Перечень документов на разработку и решаемые задачи:

- техническое задание;
- пояснительная записка;

- текст программы.

Графический материал:

- постановка задачи (плакат А1);
- схема взаимодействия программ (чертеж А1);
- диаграмма развертывания (плакат А1);
- диаграмма классов (плакат А1);
- результаты испытаний (плакат А1);
- схема алгоритма (плакат А1).

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

3 СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Организация системы

Архитектура системы разработана по модульному принципу [7]. В системе можно выделить 5 основных модулей:

- мобильный робот;
- цифровая камера;
- подсистема позиционирования;
- подсистема управления мобильным роботом;
- подсистема построения пройденного пути.

Существует две причины обоснования данного выбора: снижение требований к аппаратным узлам и более широкая возможность модификации отдельных частей.

Конечный вид существующей системы отражен на плакате «Разработка программных средств позиционирования мобильного робота. Диаграмма развертывания» шифр «11 92 00», где показана разработанная в рамках данного дипломного проекта подсистема позиционирования мобильного робота и подсистема построения траектории пройденного пути. Разрабатываемые подсистемы интегрируются в существующую систему управления мобильным роботом и являются ее логической частью. Подсистема построения траектории пройденного пути выполняет функции построения траектории пути, полученного с помощью подсистемы детектирования. В свою очередь подсистема детектирования переводит положение робота в координаты, получая изображение с аппаратно-программного модуля цифровой камеры, и переводит координаты в представление необходимое для работы модуля управления мобильным роботом. Подсистема мобильного робота при его передвижении посылает данные с одометров подсистеме управления мобильным роботом, который пересылает их в подсистему позиционирования. Данные учитываются при построении траектории пройденного пути, а так же используются при выезде робота за пределы видимости камеры.

3.2 Описание модулей общей системы

Для реализуемой системы компоненты изображенные на плакате «Разработка программных средств позиционирования мобильного робота. Диаграмма развертывания» шифр «11 92 00» имеют следующее представление:

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

- 1) Модуль мобильного робота представляет прошивку для платы коммутации Arduino, производит инициализацию логических элементов робота, управляет периферией робота, считывает и обрабатывает данные сенсоров. Получает команды воздействия по средствам Bluetooth. Передаёт данные об одометрах подсистеме управления мобильным роботом.
- 2) Модуль системы управления мобильным роботом получает данные с одометров и передаёт их модулю подсистемы позиционирования. Данный модуль получает задание от пользователя или подпрограммы испытаний и в соответствии с полученными командами производит управляющее воздействие мобильным роботом по средствам беспроводной связи Bluetooth по средствам физического протокола передачи данных RS232 семейства протоколов UART (Universal asynchronous receiver/transmitter). Команды передаются в виде букв кодировки ASCII (American Standard Code for Information Interchange). Список возможных команд представлен в таблице 3.1.

Таблица 3.1 — ASCII коды управления роботом

Задаваемое действие роботу	ASCII	Код
Передвижение робота вперед	w	119
Передвижение робота назад	x	120
Передвижение робота влево	a	097
Передвижение робота вправо	d	100
Остановка робота	s	115
Произвести инициализацию колесной базы	p	112
Подать сигнал если периферия проинициализирована	t	116

- 3) Модуль подсистемы позиционирования получает асинхронные потоки данных с видеокамеры и одометров. Производя анализ видеоизображения подсистема определяет координаты робота и захватывает объект робота. При изменении координат робота подсистема делает их запись в журнал перемещений. При получении данных с одометров система сравнивает полученный результат с камерой и записывает пройденное расстояние. Таким образом с помощью одометров производится уточнения данных полученных путём анализа видеоизображения.
- 4) Модуль цифровой камеры, рассматривается как асинхронный поток информации, имеющий форму кадров.
- 5) Модуль подсистемы построения пройденного пути использует журнал передвижения для построения карты пройденного пути.

3.3 Обзор библиотеки Qt

Qt — кросс-платформенный инструментарий разработки ПО на языке программирования C++. Существуют также привязки ко многим другим языкам программирования: Python — PyQt, PySide; Ruby — QtRuby; Java — Qt Jambi; PHP — PHP-Qt и другие [8].

Позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и Extensible Markup Language (XML). Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Существуют версии библиотеки для Microsoft Windows, систем класса UNIX с графической подсистемой X11, iOS, Android, Mac OS X, Microsoft Windows CE, встраиваемых Linux-систем и платформы S60.

Начиная с версии 4.5 Qt распространяется по 3 лицензиям:

- 1) Qt Commercial — для разработки ПО с собственной лицензией, допускающая модификацию самой Qt без раскрытия изменений.
- 2) GNU General Public License (GPL) — для разработки ПО с открытыми исходниками распространяемыми на условиях GNU GPL.
- 3) GNU Lesser General Public License (LGPL) — для разработки ПО с собственной лицензией, но без внесения изменений в Qt.

Это даёт возможность разработчику самому выбирать лицензию своего проекта, тем самым давая возможность разрабатывать как проприетарное ПО так и ПО с открытым исходным кодом.

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода. MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий мета-объектный код.

Qt позволяет создавать собственные плагины и размещать их непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Qt комплектуется визуальной средой разработки графического интерфейса Qt Designer, позволяющей создавать диалоги и формы в режиме WYSIWYG (What You See Is What You Get). В поставке Qt есть Qt Linguist — графическая утилита, позволяющая упростить локализацию и перевод вашей программы на многие языки; и Qt Assistant — справочная система Qt, упрощающая работу с документацией по библиотеке, а также позволяющая создавать кросс-платформенную справку для разрабатываемого на основе Qt ПО. Начиная с версии 4.5.0 в комплект Qt включена среда разработки Qt Creator, которая включает в себя редактор кода, справку, графические средства Qt Designer и возможность отладки приложений. Qt Creator может использовать GCC или Microsoft VC++ в качестве компилятора и GDB в качестве отладчика. Для Windows версий библиотека комплектуется компилятором, заголовочными и объектными файлами MinGW.

Библиотека разделена на несколько модулей, для четвёртой версии библиотеки это:

- 1) QtCore — классы ядра библиотеки, используемые другими модулями.
- 2) QtGui — компоненты графического интерфейса.
- 3) QtNetwork — набор классов для сетевого программирования. Поддержка различных высокоуровневых протоколов может меняться от версии к версии. В версии 4.2.x присутствуют классы для работы с протоколами File Transfer Protocol (FTP) и HyperText Transfer Protocol (HTTP). Для работы с протоколами TCP/IP (Transmission Control Protocol / Internet Protocol) предназначены такие классы, как QTcpServer, QTcpSocket для TCP и QUdpSocket для User Datagram Protocol (UDP);
- 4) QtOpenGL — набор классов для работы с OpenGL.
- 5) QSql — набор классов для работы с базами данных с использованием языка структурированных запросов Structured Query Language (SQL). Основные классы данного модуля в версии 4.2.x: QSqlDatabase — класс для предоставления соединения с базой, для работы с какой-нибудь конкретной базой данных требует объект, унаследованный от класса QSqlDriver — абстрактного класса, который реализуется для конкретной базы данных и может требовать для компиляции Software Development Kit (SDK) базы данных. Например, для сборки драйвера под базу данных FireBird/InterBase требует .h файлы и библиотеки статической линковки, входящие в комплект поставки данной БД.
- 6) QtScript — классы для работы с Qt Scripts.
- 7) QtSvg — классы для отображения и работы с данными Scalable Vector Graphics (SVG).
- 8) QtXml — модуль для работы с XML, поддерживается Simple API for XML (SAX) и Document Object Model (DOM) модели работы.
- 9) QtDesigner — классы создания расширений QtDesigner'а для своих собственных виджетов.
- 10) QtUiTools — классы для обработки в приложении форм Qt Designer.

QtAssistant — справочная система.

11) Qt3Support — модуль с классами, необходимыми для совместимости с библиотекой Qt версии 3.x.x.

12) QtTest — модуль для работы с UNIT тестами.

13) QtWebKit — модуль WebKit, интегрированный в Qt и доступный через её классы.

14) QtXmlPatterns — модуль для поддержки XQuery 1.0 и XPath 2.0.

15) Phonon — модуль для поддержки воспроизведения и записи видео и аудио, как локально, так и с устройств и по сети.

16) QtCLucene — модуль для поддержки полнотекстового поиска, применяется в новой версии Assistant в Qt 4.4.

17) ActiveQt — модуль для работы с ActiveX и Component Object Model (COM) технологиями для Qt-разработчиков под Windows.

18) QtDeclarative — модуль, предоставляющий декларативный фреймворк для создания динамичных, настраиваемых пользовательских интерфейсов.

Библиотека использует собственный формат проекта, именуемый .pro файлом, в котором собрана информация о том, какие файлы будут скомпилированы, по каким путям искать заголовочные файлы и много другой информации. Впоследствии при помощи утилиты qmake из них получаются makefile для make-утилиты компилятора. Также есть возможность работы при помощи интеграторов с Microsoft Visual Studio 2003/2005/2008/2010. Так же доступна интеграция в Eclipse для версии библиотеки 4.x.x.

3.4 Структура Qt и использованные классы

Структурные типы данные [9]:

1) QImage — аппаратно-независимое представление изображения, предоставляющее прямой доступ к пикселям и способная работать в качестве устройства рисования.

2) QPixmap — неэкранный представление изображения, которое может использоваться в качестве устройства рисования.

3) QSettings — обеспечивает платформу-независимые настройки приложения, являясь абстракцией вокруг различных технологий хранения настроек приложений, а так же поддерживает пользовательские форматы хранения данных.

4) QTimer — предоставляет интерфейс высокого уровня для повторяющихся и однократных таймеров.

5) QWidget — базовый класс для всех объектов пользовательского интерфейса. Является частью пользовательского интерфейса: он обрабатывает события мыши и клавиатуры, других событий оконного окружения, цветовой гаммы.

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

6) QEvent — базовый класс для всех объектов событий. Объект события содержит параметры события.

7) QTimerEvent — содержит параметры которые описывает событие таймера.

8) timerEvent — обработчик события таймера. Переопределяется для создания своего события с заданной частотой. Запускается методом startTimer(), в параметре которого указывается через сколько миллисекунд будет выполняться обработчик.

9) QMainWindow — предоставляет главное окно приложения. Главное окно предоставляет структуру для создания пользовательского интерфейса приложения. Qt имеет класс QMainWindow и связанные с ним классы для управления главным окном. QMainWindow имеет собственный компоновщик, в который можно добавлять QToolBar'ы, QDockWidget'ы, QMenuBar, и QStatusBar. Компоновщик имеет центральную область, которая может быть занята любым виджетом.

10) QMenuBar — предоставляет интерфейс горизонтального меню.

11) QAction — класс обеспечивает обработку абстрактных действий с пользовательским интерфейсом, который может привязан к любому виджету.

12) QMenu — класс предоставляет виджет меню, который возможно использовать как панель-меню, контекстное-меню или как всплывающее меню.

13) QStatusBar — класс реализует горизонтальную панель, приспособленную для предоставлении информации о состоянии.

14) QComboBox — класс предоставляет графический элемент кнопки с выпадающим списком.

15) QLabel — класс предоставляет графический элемент размещающий на форме текстовый элемент с задаваемым задним фоном как в виде цвета, так и в виде изображения.

16) QCheckBox — класс предоставляет графический элемент флажок с текстовой меткой.

17) QHBoxLayout — класс предоставляет выравнивающий контейнер для графических элементов, помещенных в него.

18) QSlider - класс предоставляет графический элемент горизонтального бегунка.

19) QLineEdit — класс предоставляет графический элемент строки для редактирования текстовой информации.

20) QPushButton — класс предоставляет графический элемент кнопки.

21) QWidget — класс предоставляет набор виджетов со вкладками.

Механизм сигналов и слотов играет решающую роль в разработке программ Qt. Он позволяет прикладному программисту связывать различные объекты, которые ничего не знают друг о друге. Слоты почти совпадают с обычными функциями, которые объявляются внутри классов C++ (функции-члены). Они делятся на виртуальные, перегруженные, открытые (public), защищенные (protected) и закрытые (private), они могут вызываться непосредственно, как и любые другие функции-члены C++. Их параметры, могут быть любого типа. Слоты, в отличие от обычных функций-членов,

могут подключаться к сигналам, и в результате они будут вызываться при каждом генерировании соответствующего сигнала.

Приложения с графическим интерфейсом управляются событиями: все, что происходит в приложении — есть результат обработки тех или иных событий. При разработке программ под Qt, необходимость использовать механизм событий возникает редко, поскольку виджеты Qt выдают сигналы, когда происходит нечто значительное. События приобретают значение в том случае, когда необходимо создать новый виджет или когда нужно расширить функциональность существующего виджета.

События генерируются оконной системой или Qt, в ответ на различные ситуации. Когда нажимается или отпускается клавиша на клавиатуре или кнопка мыши, генерируется соответствующее событие. Когда перемещается одно окно и в результате этого перемещения открывается другое, лежавшее ниже, возникает событие, которое сообщает открывшемуся окну о необходимости перерисовать себя. События генерируются всякий раз, когда виджет теряет или получает фокус ввода. В большинстве своем, события генерируются в ответ на действия пользователя, но иногда, например события от таймера, они генерируются системой независимо от пользователя.

3.5 Обзор библиотеки OpenCV

OpenCV (Open Source Computer Vision Library) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, так же разрабатывается для Python, Ruby, Matlab. Может свободно использоваться в академических и коммерческих целях — распространяется в условиях лицензии BSD (Berkeley Software Distribution).

Библиотека состоит из 5 модулей (модуль `sxcore` является основным и называется ядром библиотеки), каждый из которых реализует определенный класс функциональности [10].

Ядро `sxcore` реализует следующую функциональность:

- 1) базовые операции над многомерными числовыми массивами;
- 2) матричная алгебра, математические функции, генераторы случайных чисел;
- 3) базовые функции 2D графики;
- 4) поддержка более сложных структур данных: разреженные массивы, динамически растущие последовательности, графы.

Модуль `cv` предоставляет функциональность для обработки изображений и компьютерного зрения:

- 1) базовые операции над изображениями (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.);

- 2) анализ изображений (выбор отличительных признаков, морфология, поиск контуров, гистограммы);
- 3) структурный анализ (описание форм, плоские разбиения);
- 4) анализ движения, слежение за объектами;
- 5) обнаружение объектов, в частности лиц;
- 6) калибровка камер, элементы восстановления пространственной структуры.

Модуль `highgui` предназначен для ввода/вывода изображений и видео, создания пользовательского интерфейса:

- 1) захват видео с камер и из видео файлов, чтение/запись статических изображений;
- 2) функции для организации простого интерфейса пользователя (сейчас все демо приложения используют `HighGUI`).

`SvauX` – модуль содержащий экспериментальные и устаревшие функции:

- 1) объемное зрение: стерео калибровка, само калибровка;
- 2) поиск стерео соответствия, клики в графах;
- 3) нахождение и описание черт лица;
- 4) сравнение форм, построение скелетонов;
- 5) скрытые Марковские цепи;
- 6) описание текстур.

`CvCam` — модуль для захвата видео, на данный момент поддержка прекращена.

Если же говорить о том, что находится в структурах данных `OpenCV`, то первое, о чем следует упомянуть, это, собственно, изображения. Поскольку библиотека создана лабораторией Intel, нет ничего удивительного в использовании для этих целей `IplImage` — формата, заимствованного из Intel Image Processing Library (IPL). В `IplImage` инкапсулируются другие стандартные форматы изображений. Помимо глобальных операций по созданию нового изображения или удалению существующего, есть несколько макросов попиксельного редактирования.

Структура `OpenCV` и использованные сущности

В разрабатываемой системе используются четыре модуля библиотеки `OpenCV` `cv`, `cvcam`, `cxcore` и `highgui`, описанных в разделе «Состав, структура и назначение системы» в подразделе «Обзор `OpenCV`». В рамках данного проекта используется только часть доступной функциональности этих модулей. Проведем обзор использованных типов данных и методов библиотеки `OpenCV`.

Структурные типы данных:

- 1) `IplImage` – хранит изображение, и необходимую информацию о нем.
- 2) `CvMemStorage` – организует динамический список для хранения данных.
- 3) `CvSeq` – хранит сегменты данных в виде динамического списка.
- 4) `CvCapture` – используется, для хранения адреса видеозахватывающего устройства.

3.6 Обзор платформы для управления аппаратным обеспечением Arduino

Arduino — аппаратная вычислительная платформа, основными компонентами которой являются простая плата ввода/вывода и среда разработки на языке Processing/Wiring. Arduino может использоваться как для создания автономных интерактивных объектов, так и подключаться к программному обеспечению, выполняемому на компьютере (например, Macromedia Flash, Processing, Max/MSP, Pure Data, SuperCollider).

Плата Arduino состоит из микроконтроллера Atmel AVR (ATmega328 и ATmega168 в новых версиях и ATmega8 в старых), а также элементов обвязки для программирования и интеграции с другими схемами. На многих платах присутствует линейный стабилизатор напряжения +5 или +3,3 В. Тактирование осуществляется на частоте 16 или 8 МГц кварцевым резонатором. В микроконтроллер предварительно прошивается загрузчик BootLoader, поэтому внешний программатор не нужен.

На концептуальном уровне все платы программируются через RS-232 (последовательное соединение), но реализация этого способа отличается от версии к версии. Плата Serial Arduino содержит простую инвертирующую схему для конвертирования уровней сигналов RS-232 в уровни ТТЛ, и наоборот. Текущие рассылаемые платы, например, Diecimila, программируются через USB, что осуществляется благодаря микросхеме конвертера USB-to-Serial FTDI FT232R.

Платы Arduino позволяют использовать большую часть I/O выводов микроконтроллера во внешних схемах. Эти сигналы доступны на плате через контактные площадки или штыревые разъемы. Также доступны несколько видов внешних плат расширения, называемых «shields» («шилды»), которые присоединяются к плате Arduino через штыревые разъемы. Одной из таких плат является bluetooth, по средствам которого можно реализовывать беспроводную связь с контроллером.

Интегрированная среда разработки Arduino это кроссплатформенное приложение на Java, включающее в себя редактор кода, компилятор и модуль передачи прошивки в плату.

Среда разработки основана на языке программирования Processing и спроектирована для программирования новичками, не знакомыми близко с разработкой программного обеспечения для электроники. Язык программирования аналогичен используемому в проекте Wiring. Строго говоря, это C++, дополненный некоторыми библиотеками. Программы обрабатываются с помощью препроцессора, а затем компилируется с помощью AVR-GCC.

Arduino оптимально подходит для поставленной задачи так как обладает следующими качествами:

- управление низкоуровневой периферией;

- возможность удаленного управления;
- низкий порог вхождения в программирование микроконтроллеров;
- мощное средство для разработки прошивки — Arduino IDE;
- кроссплатформенность и переносимость;
- распространяется под открытой лицензией.

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

4 ОПИСАНИЕ РЕАЛИЗАЦИИ ПРОГРАММНОЙ СИСТЕМЫ И АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ

4.1 Описание среды разработки Qt Creator

В качестве среды разработки дипломного проекта была использована Qt Creator 2.1. Эта среда разработки предназначена для разработки надежных многоуровневых кроссплатформенных приложений на C++, Python и JavaScript для Windows, Linux Mac OS, мобильных и других устройств, на которых возможна установка библиотек Qt.

Qt Creator 2.1 предоставляет мощную и гибкую среду разработки для создания приложений на базе библиотек QtLibs. Ее можно использовать в качестве интегрированной системы разработки или в качестве набора отдельных средств. Разработчикам Qt Creator предлагает следующие преимущества:

- 1) Расширенные возможности создание приложений с помощью включенных библиотек фреймворка Qt.
- 2) Создание точно настраиваемых приложений и компонентов, использующих на основе графической библиотеки QtLibs.
- 3) Использование среды управляемых расширений C++ и поддержки компилятора для создания оптимизированного кода. Встраивание компонентов Qt Framework, включая сборку мусора, Qt Forms и использование потоков.
- 4) Создание точно настраиваемых неуправляемых приложений и компонентов.
- 5) Создание и компиляция полностью неуправляемого кода для версии x86, x86_64, arm. Повышение производительности приложения и сокращение его объема с помощью средств оптимизации для целого ряда процессоров, включая средство оптимизации программы в целом и поддержку для наборов инструкций SSE (Streaming SIMD Extensions) и SSE2 (Streaming SIMD Extensions 2).
- 6) Для компиляции существующего кода C++ для среды Qt Creator не требуется переписывать его на новом языке. Qt Creator поддерживает смешанный код (управляемый и неуправляемый) и смешанные данные, обеспечивая максимальную производительность и степень контроля.
- 7) Разработка с соблюдением отраслевого стандарта. Создание современного кода на C++ и исходных кодов библиотек с помощью компилятора, соблюдающего строгие требования стандарта ISO C++.
- 8) В Qt Creator выполняется компиляция современного кода C++, в котором используются расширенные функции шаблонов, включая частичную специализацию шаблонов и частичное упорядочивание шаблонов функций. Использование распространенных библиотек, созданных сообществом, включая библиотеку Boost.

9) Использование усовершенствованных библиотек для внедрения расширенных функций.

10) Современный компилятор gcc и компоненты языка, делающие создание сложного кода простым и безопасным процессом.

11) Экономное создание кода в расширяемой среде IDE.

4.2 Выделение классов и взаимосвязей между ними

Для реализации разрабатываемой системы выбран объектно-ориентированный подход программирования. При анализе поставленной задачи были выделены следующие типы объектов: датчик расстояния, одометры, робот, цифровая камера, анализатор пройденного пути, система управления мобильным роботом. Данные объекты, с позиции разрабатываемой системы, представлены в виде циклически поступающих и отправляемых данных, для цифровой камеры, одометров, анализатора пройденного пути и модуля управления роботом соответственно. Это означает, что общая организация системы будет выглядеть, как цикл, состоящий из следующих фаз:

- 1) получение данных от объекта;
- 2) обработка данных;
- 3) передача данных объекту системы управления роботом и анализатору пройденного пути.

Объект типа «одометры» наследуется от виртуального класса датчик расстояния. При инициализации объекта типа одометр используется одно из основополагающих свойств ООП — наследование. Такой подход позволяет расширять возможности системы, добавляя в неё новые датчики расстояния, дополняя функционал родительского класса.

Объект типа «робот» представляет собой класс, который абстрагирует процесс управления роботом, является обёрткой над стандартным протоколом передачи данных. Его методами являются действия, которые робот должен выполнить.

Объект типа «анализатор пройденного пути» использует журнал пройденного пути для построения траектории, по которой двигался робот. Совмещает данные одометров и координат робота полученных по средствам анализа видеоизображения. В результате получаем пройденный путь с возможностью измерить расстояние на выбранном отрезке пути.

Объект типа «система управления мобильным роботом» содержит метод для тестирования, который отвечает за испытание системы. При помощи Bluetooth связи объект посылает управляющие команды роботу на исполнение.

Поскольку для получения данных от объекта видекамера используется функциональность модуля cvcam, библиотеки OpenCV, процесс захвата

видеоизображения представлен автоматическим вызовом функции, при помощи класса QTimer. В случае использования данной технологии в системах реального времени, каждая итерация обработки информации должна занимать строго фиксированный квант времени. В данной системе на одну итерацию отводится не более чем 30 миллисекунды, этот параметр связан с частотой формирования кадров видеоряда цифровой камерой составляющей 30 кадров в секунду. Для реализации работы с объектом камера, на основе функциональности OpenCV, разработан класс ImageCapture реализующий описанный цикл функционирования системы. Этот класс реализует методы для захвата данных от объекта камера, а так же, для удобства тестирования и демонстрации результатов, обеспечивает возможность работы с видеофайлами формата mpeg.

4.3 Описание использованных алгоритмов

Для решения поставленной задачи, были реализованы следующие алгоритмы обработки и получения данных, алгоритмов передвижения:

- 1) Калибровка видеокамеры
- 2) Детектирование паттерна робота
- 3) Получение данных с одометра

Калибровка камеры выполняется после инициализации системы и требуется для корректной работы системы. Производится пользователем и состоит из следующих этапов:

- 1) Калибровка фокусных искажений камеры;
- 2) Калибровка распознавания цвета.

Калибровка фокусных искажений производится с помощью стандартных методов библиотеки OpenCV. Для этого на край изображения, захватываемого камерой располагается калибровочный паттерн (см. рисунок 4.1) и подбираются значения матриц искажения для компенсации эффекта «бочки». Нужно отметить что большинство камер среднего класса и выше не требуют такой калибровки, но бюджетные модели вносят значительные искажения, которые существенно влияют на результаты детектирования и навигации.

Калибровка цветowych фильтров выполняется по средствам ручного подбора диапазона по следующему алгоритму:

- 1) Выбирается цвет для калибровки.
- 2) Получение фотографии с камеры.
- 3) Перевод изображения в цветовую палитру HSV.
- 4) Загрузка сохраненных параметров диапазона выбранного цвета. Подстройка диапазона каждой составляющей палитры HSV. Цвет попадающий в

диапазон отображается белым цветом на изображении, не попавший — черным.

5) Сохранение новых параметров.

Необходимо отметить что калибровка цветов должна производиться при значительном изменении параметров окружающей среды, что позволит системе работать более стабильно. Неравномерное освещение слабо влияет на качество распознавания, благодаря использованию палитры HSV.

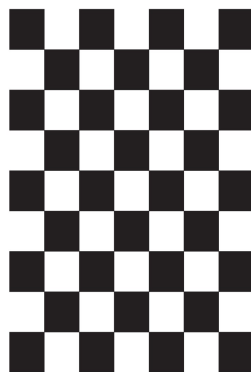


Рисунок 4.1 — Паттерн для калибровки фокусных искажений видеокамеры

Для детектирования паттерна робота создано два алгоритма. Первый алгоритм работает для паттерна, изображенного на рисунке 4.2, и служит для испытаний системы, а второй на рисунке 4.3, используется в соревнованиях Robocup Soccer Small League и используется в окружающей среде с множеством объектов.



Рисунок 4.2 — Паттерн для испытаний

Для первого паттерна реализован следующий алгоритм:

- 1) Выделение контуров на полученном изображении.
- 2) Поиск прямоугольников среди полученных контуров.
- 3) Выделяется Rectangle Of Interests (ROI), с радиусом на 20% больше чем размер паттерна робота.

- 4) Определяется центр масс паттерна.
- 5) Вычисления угла на основе метки паттерна.



Рисунок 4.3 — Паттерн для определения нескольких объектов

Паттерн Robocup Small League распознается по следующему алгоритму:

- 1) Определяется центральный желтый круг паттерна.
- 2) Выделяется Rectangle Of Interests (ROI), с радиусом на 20% больше чем размер паттерна робота (см рисунок 4.4).
- 3) Определяются вспомогательные цвета.
- 4) Определяется центр масс паттерна.
- 5) Определяется вектор направления робота (см. рисунок 4.5)

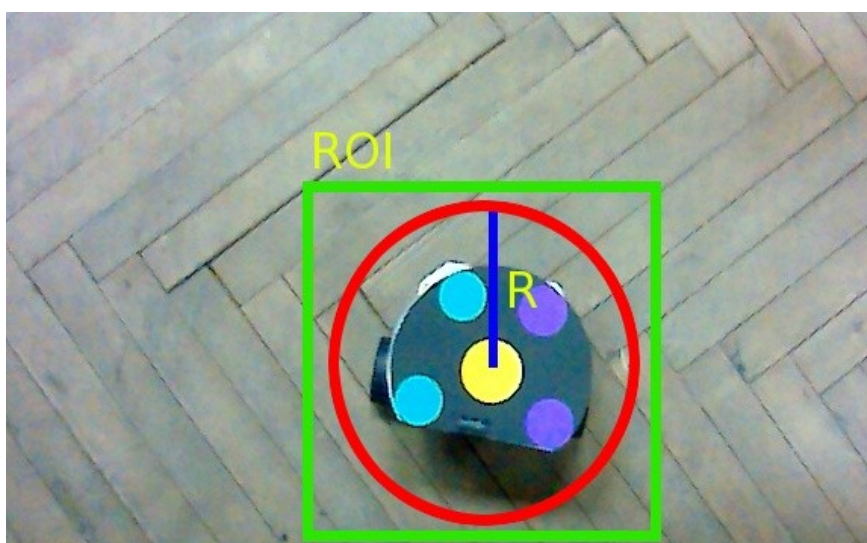


Рисунок 4.4 — Паттерн для калибровки фокусных искажений видеокамеры

Для повышения надежности работы алгоритма детектирования робота, применяется следящее окно на основе полученного ROI, представленное ограниченной частью изображения с находящимся на нем роботом. Следящее окно перемещается при передвижении робота, согласно смещению его центра масс.



Рисунок 4.5 — Паттерн для калибровки фокусных искажений видеокамеры

Алгоритм получение данных с одометров:

- 1) Данные о состоянии одометра считываются каждые 30мс.
- 2) При изменении состоянии одометра и заданной команде «Вперед» робот увеличивает счетчик.
- 3) При изменении состоянии одометра и заданной команде «Назад» робот увеличивает счетчик.
- 4) Полученные данные робот переводит в расстояние по следующей формуле (4.1).

$$L_{\text{пройдено}} = \frac{\text{Количество пройденных меток}}{\text{Количество меток колеса}} * 2 * \pi * R \quad (4.1)$$

где π — число пи,
 R — радиус колеса робота.

4.4 Описание реализованных классов

При реализации системы были получены следующие классы:

- 1) RobotControl — реализует управление роботом, хранит данные о его положении и направлении;
 LenthSensor — абстрактный класс для датчиков расстояния;
- 2) OdomentSensor — хранит данные получаемые с одометра;

- 3) CameraCalibrate — реализует настройку камеры;
- 4) PatternRecognition — реализует алгоритмы распознавания паттернов;
- 5) ImageCapture — производит захват изображения из видеопотока, получаемого из видеофайла или от цифровой видеокамеры;
- 6) Mapping — производит построение траектории пройденного пути;
- 7) Navigation — производит определение координат робота на основе имеющихся датчиков и камеры.

Диаграмма классов представлена на плакате «Разработка программных средств позиционирования мобильного робота. Диаграмма классов» шифр «11 93 00».

					ДП.АС24.06791 – 11 81 00	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

5 РЕЗУЛЬТАТЫ ИСПЫТАНИЙ СИСТЕМЫ

Для проверки корректности работы были разработаны следующие группы испытаний:

- 1) детектирование робота;
- 2) передвижение робота из точки А в точку Б;
- 3) передвижения робота по маршруту точек А, Б и остановке в точке В;

Для наглядности процессов испытаний используется графическое представление результатов.

Первая группа предназначена для тестирования алгоритма детектирования робота на изображении. Для проведения испытания, в системе позиционирования выбирается паттерн Robocup Soccer и включается испытание детектирования для текущей ситуации (см. рисунок 5.1).



Рисунок 5.1 — Кадр, содержащий изображение мобильного робота

После запуска системы был выбран паттерн для распознавания — паттерн Robocup Soccer Simulation League. Для повышения точности детектирования была произведена калибровка камеры, настроены белый, черный, зеленый, желтый, фиолетовый цвета. Для испытания использовался робот футболист с серводвигателями ведущих колес.

Результат испытания изображен в виде линии определяющей ориентацию робота в пространстве. После детектирования на изображении эта линия отображается синим цветом, что свидетельствует об успешном распознавании (см. рисунок 5.2).

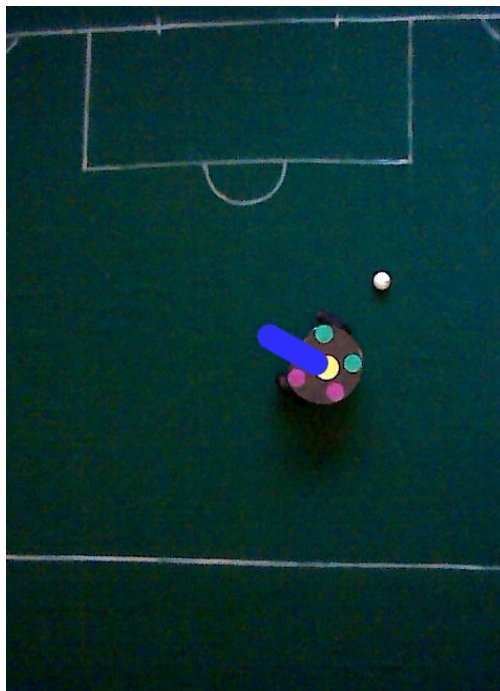


Рисунок 5.2 — Кадры, после детектирования

Для испытания системы второй группой тестов системе позиционирования задаётся точка Б, куда необходимо привести робота. Было проведено испытание системы. Начальная позиция робота отображена на рисунке 5.3.

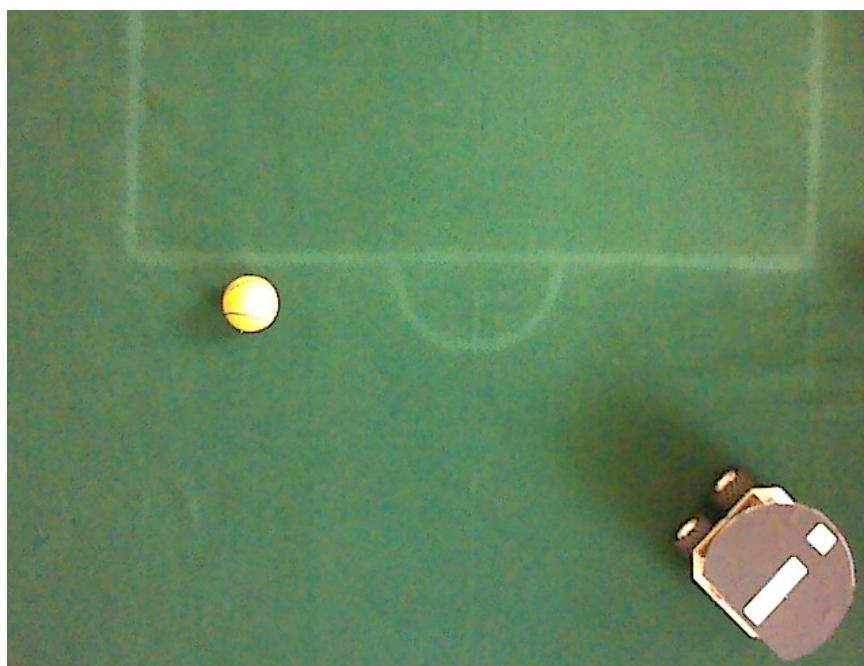


Рисунок 5.3 — Начальное положение робота

После запуска системы первой итерацией робот развернулся в сторону мяча (см. рисунок 5.4).

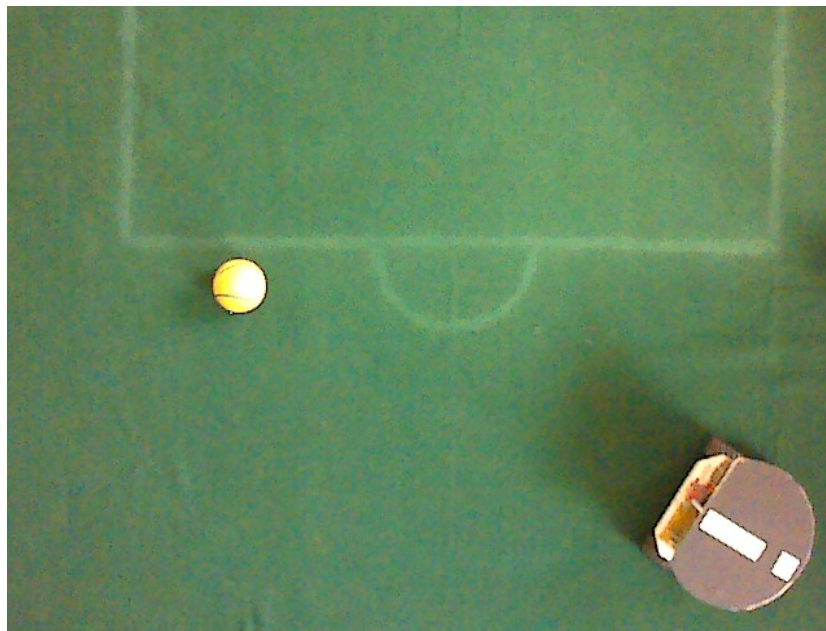


Рисунок 5.4 — Первая итерация испытания, разворот робота

Прибытие робота в контрольную точку (к мячу) отображено на рис 5.5. Таким образом мобильный робот выполнил поставленное испытание.

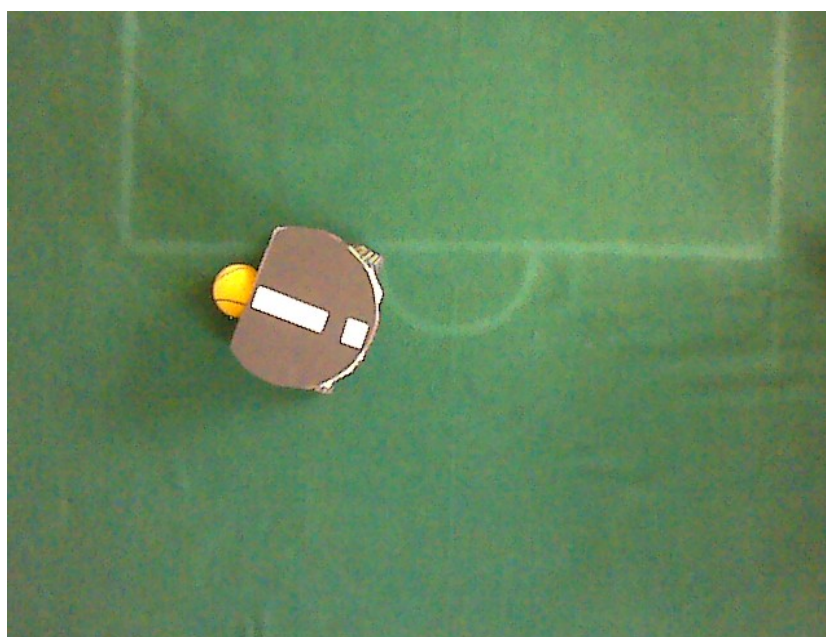


Рисунок 5.5 — Робот прибыл в контрольную точку к мячу

Для проведения тестов третьей группы необходимо задать на изображении контрольных точек для прохождения и запустим систему. Результат тестов на рисунках 5.6 — 5.11.

Первым этапом является детектирование робота на изображении. Когда робот найден определяется его направление. Далее строится линия соединяющие робота и контрольную точку. Далее определяется угол между направлением робота и

построенной линии. Если угол более 10 градусов робот разворачивается. В данном тесте начальное направление робота не требует его разворота (см рисунок 5.6), поэтому робот сразу направляется в первую контрольную точку и при достижении заданного в системе радиуса, останавливается (см. рисунок 5.7).

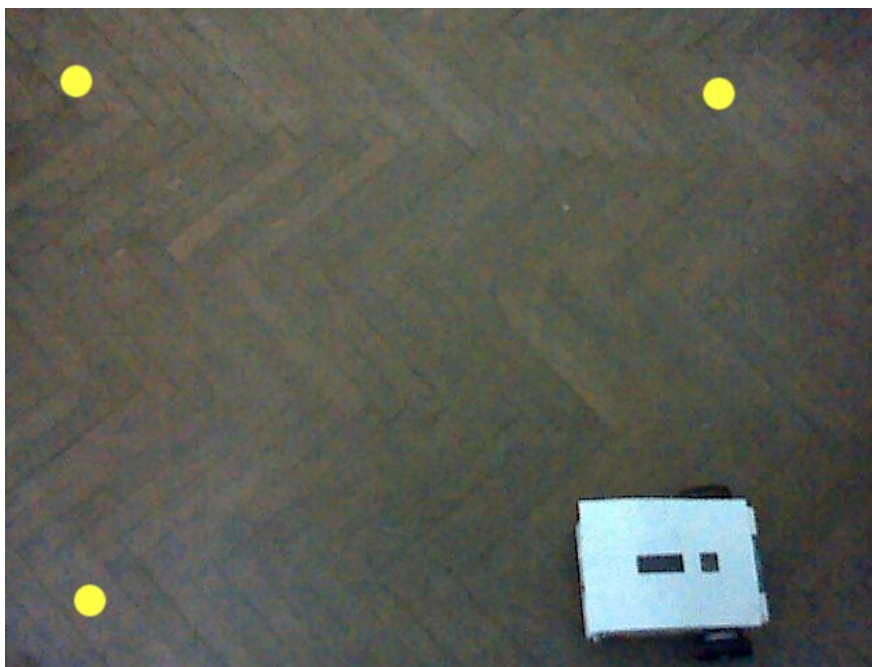


Рисунок 5.6 — Начальное положение робота, задание контрольных точек роботу

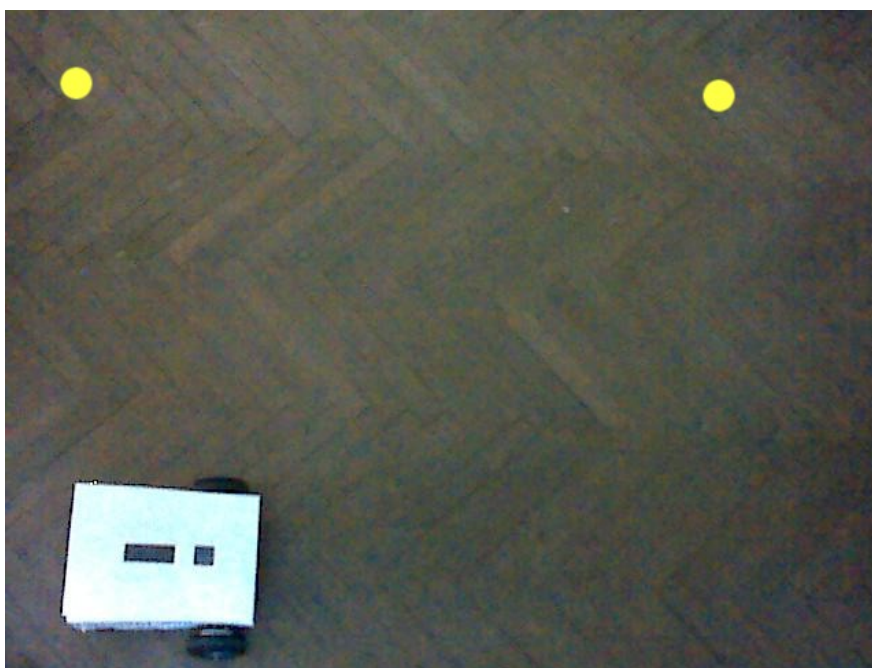


Рисунок 5.7 — Первая итерация, робот приехал к первой контрольной точке

Достигнув первую контрольную точку робот начал разворот в сторону второй контрольной точки. Робот закончил маневр разворота когда его направление соответствовало второй контрольной точке (см. рисунок 5.8).

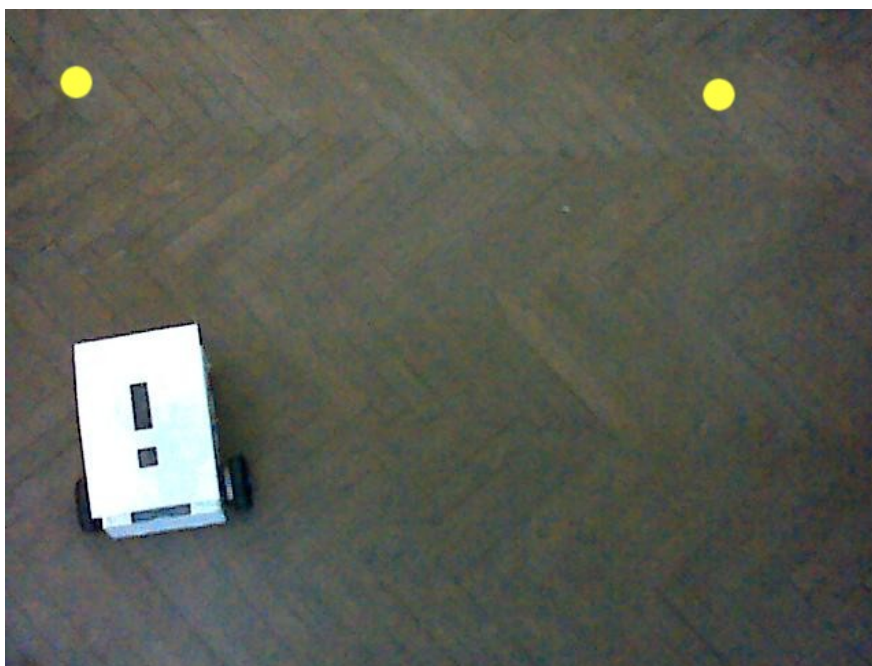


Рисунок 5.8 — Робот после завершения маневра поворота

Далее робот переместился ко второй контрольной точке (см. рисунок 5.9). По достижению установленного в системе минимально радиуса робот остановился возле контрольной точки.

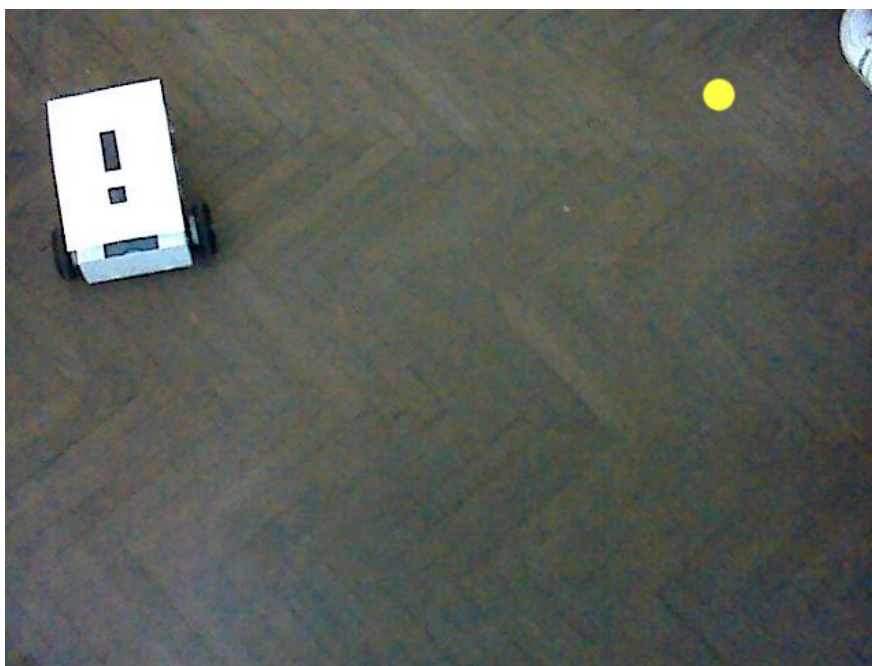


Рисунок 5.9 — Робот достиг второй контрольной точки

Робот совершил маневр поворота в сторону третьей контрольной точки и остановился когда угол по направлению к этой точке стал меньше 10 градусов (см. рисунок 5.10). После этого он начал передвижение.

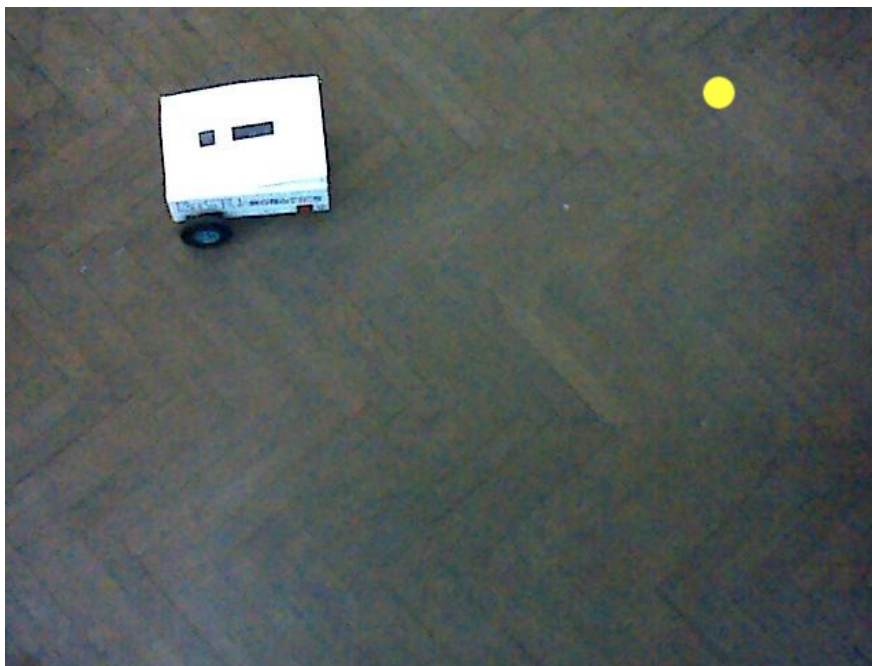


Рисунок 5.10 — Результат маневра разворота по направлению к третьей контрольной точке

Робот остановился рядом с третьей контрольной точкой тем самым успешно завершив третью группу испытаний (см. рисунок 5.11).



Рисунок 5.11 — Робот достиг третьей контрольной точки

В результате проведения третьей группы испытаний были получены обработанные данные одометров и построен график пройденного пути (см. рисунок 5.12).

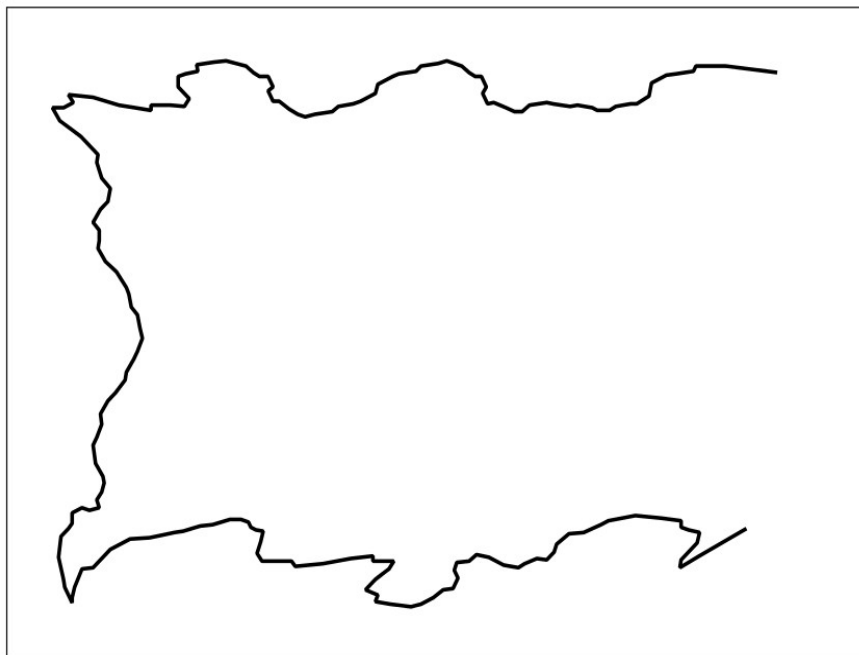


Рисунок 5.12 — Робот достиг третьей контрольной точки

Анализ полученных при испытаниях результатов показывает, что система работает без детектирования ложных областей в качестве объектов. Искажение, полученное при детектировании компенсируется калибровкой камеры, вызвано наличием естественного освещения сцены и искажений камеры. Система навигации успешно детектировала различные паттерны на роботах, а так же провела робота по заданным контрольным точкам.

6 РАСЧЕТ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ

6.1 Расчет общей трудоемкости разработки программного обеспечения

Среда разработки ПО — Qt Creator 2.1. ПО функционального назначения.

Объем программного средства (ПС) определяется на основе данных, приведенных в таблице 6.1.

Таблица 6.1 — Перечень и объем функций программного обеспечения

Код функции	Наименование (содержание) функции	Объем функции строк исходного кода (ЛОС)	
		По каталогу V_i	Уточненный V_{yi}
101	Организация ввода информации	110	110
102	Контроль, предварительная обработка и ввод информации	340	270
107	Организация ввода/вывода информации в интерактивном режиме	220	220
405	Система настройки ПО	300	240
503	Управление внешними устройствами и объектами	6340	3200
602	Вспомогательные и сервисные программы	490	120
707	Графический вывод результатов	330	100
ИТОГО		8130	4260

Общий объем ПС рассчитывается по формуле:

$$V_o = \sum_{i=1}^n V_i, \quad (6.1)$$

где V_o — общий объем ПС;

V_i — объем функций ПС;

n — общее число функций.

Подставляя исходные данные, получим:

$$V_0 = 110 + 340 + 220 + 300 + 6340 + 490 + 330 = 8130 \text{ (условных машинных команд)}$$

Уточненный объем ПС рассчитывается по формуле:

$$V_y = \sum_{i=1}^n V_{yi} \quad (6.2)$$

где V_y — уточненный объем ПС;

V_{yi} — уточненный объем функций ПС;

Подставляя исходные данные, получим:

$$V_{yi} = 110 + 270 + 220 + 240 + 3200 + 120 + 100 = 4260 \text{ (условных машинных команд)}$$

В связи с использованием более совершенных средств автоматизации объемы функций 102, 405, 503, 602 и 707 были уменьшены и уточненный объем ПО (V_y) составил 4260 строк исходного кода (LOC) вместо 8130.

ПО отнесено ко второй категории сложности: требования пользователей предполагают связь данного ПО с другими системами, а так же обеспечение хранения, ведения и поиска данных в сложных структурах. Наличие характеристики, определяющей сложность ПО, позволяет применить к объему ПО коэффициент K_c :

$$K_c = 1 + 0,12 = 1,12. \quad (6.3)$$

ПО является развитием определенного параметрического ряда систем-аналогов.

Новизна ПО соответствует категории Б, а $K_n = 0,72$.

При разработке ПО доля стандартных модулей составляет 65 %, а $K_T = 0,75$.

Новизне ПО категории Б соответствует следующее распределение трудоемкости по стадиям — $K_{тз} = 0,10$; $K_{эп} = 0,20$; $K_{тп} = 0,30$; $K_{рп} = 0,30$; $K_{вн} = 0,10$.

Коэффициент, учитывающий средства разработки ПО $K_{ур} = 1,0$.

Нормативная трудоемкость разработки ПО (T_n) составляет 223 чел.-дн.

На основании принятого к расчету объема (V_y) и категории сложности ПО определяется нормативная трудоемкость (T_n) по стадиям разработки.

Нормативная трудоемкость ПО (T_n) выполняемых работ по стадиям разработки может корректироваться при необходимости с помощью:

- 1) коэффициента повышения сложности ПО (K_c);
- 2) коэффициента, учитывающего новизну ПО (K_n);
- 3) коэффициента, учитывающего степень использования стандартных модулей (K_T);
- 4) коэффициента, учитывающего средства разработки ПО ($K_{ур}$).

Нормативная трудоемкость определяется по формулам (6.4 – 6.8):

1) для стадии ТЗ — проведение исследований

$$T_{y_{ТЗ}} = T_n \cdot K_{ТЗ} \cdot K_c \cdot K_n \cdot K_{ур}, \quad (6.4)$$

2) для стадии ЭП — анализ требований

$$T_{y_{ЭП}} = T_n \cdot K_{ЭП} \cdot K_c \cdot K_n \cdot K_{ур}, \quad (6.5)$$

3) для стадии ТП — проектирование

$$T_{y_{ТП}} = T_n \cdot K_{ТП} \cdot K_c \cdot K_n \cdot K_{ур}, \quad (6.6)$$

4) для стадии РП — разработка

$$T_{y_{РП}} = T_n \cdot K_{рп} \cdot K_c \cdot K_n \cdot K_t \cdot K_{ур}, \quad (6.7)$$

5) для стадии ВН — ввод в действие

$$T_{y_{ВН}} = T_n \cdot K_{вн} \cdot K_c \cdot K_n \cdot K_{ур}, \quad (6.8)$$

где $K_{ТЗ}$, $K_{ЭП}$, $K_{ТП}$, $K_{рп}$ и $K_{вн}$ — значения коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости ПО.

Таблица 6.2 – Расчет общей трудоемкости разработки ПО

Показатели	Стадии					Итого
	ТЗ	ЭП	ТП	РП	ВН	
1. Коэффициенты удельных весов трудоемкости стадии разработки ПО (K)	0,10	0,20	0,30	0,30	0,10	1,0
2. Распределение нормативной трудоемкости ПО (T_n) по стадиям, чел.-дн.	22	45	67	67	22	223
3. Коэффициент сложности ПО (K_c)	1,12	1,12	1,12	1,12	1,12	—
4. Коэффициент, учитывающий использование стандартных модулей (K_t)	—	—	—	0,55	—	—
5. Коэффициент, учитывающий новизну ПО (K_n)	0,72	0,72	0,72	0,72	0,72	—
6. Коэффициент, учитывающий средства разработки ПО ($K_{ур}$)	1	1	1	1	1	—
7. Общая трудоемкость ПО (T_o), чел.дн.	24	37	54	30	24	169

Использование коэффициентов сложности, новизны, учитывающих средства разработки и степень использования стандартных модулей позволяет определить общую трудоемкость разработки ПО, которая равна сумме нормативной (скорректированной) трудоемкости ПО по стадиям разработки:

$$T_o = \sum_{n=1}^n T_{y_i} , \quad (6.9)$$

где T_{yi} — нормативная (скорректированная) трудоемкость разработки ПО на i -й стадии (чел.-дн.);

n — количество стадий разработки.

6.2 Расчет заработной платы разработчиков программного обеспечения

Эффективный фонд времени работы одного работника ($\Phi_{эф}$) рассчитывается по формуле:

$$\Phi_{эф} = D_r - D_{п} - D_v - D_o, \quad (6.10)$$

где D_r — количество дней в году;

$D_{п}$ — количество праздничных дней в году;

D_v — количество выходных дней в году;

D_o — количество дней отпуска.

Эффективный фонд времени одного работника определяется с использованием производственного календаря за 2011 год: D_r — 365 дн, D_v — 104 дн., $D_{п}$ — 5 дн., D_o — 21 дн.

$$\Phi_{эф} = 365 - 104 - 5 - 21 = 235 \text{ (дн)}. \quad (6.11)$$

Общая плановая численность разработчиков ($Ч_{разоб}$) рассчитывается на основании уточненной трудоемкости разработки программного средства (T_y) и установленного периода разработки ($T_{пл}$):

$$Ч_{разоб} = \frac{T_{yt}}{T_{пл} * \Phi_{эф}} \quad (6.12)$$

где $Ч_{разоб}$ — общая плановая численность разработчиков (чел.);

$T_{пл}$ — плановая продолжительность разработки программного средства (лет);

					ДП.АС24.06791 – 11 81 00	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

$\Phi_{\text{эф}}$ — эффективный фонд времени работы работника в течение года (дн./год).
 $T_{\text{пл}} = 6$ месяцев или 0.5 года.

$$\mathcal{C}_{\text{разоб}} = \frac{169 \text{ чел-дн}}{0.5 \text{ года} * 235 \text{ дн}} = 1,44 \approx 2 (\text{чел}) \quad (6.13)$$

В создании ПО будет принимать участие руководитель.

Рассчитаем продолжительность участия разработчика в создании ПО и результаты оформим в виде таблицы (см. таблицу 6.3).

Таблица 6.3 — Определение продолжительности участия разработчика в создании ПО

Показатели	Стадии разработки ПО					Всего
	ТЗ	ЭП	ТП	РП	ВН	
Общая трудоемкость ПО (T_o), чел.дн.	24	37	54	30	24	169
Численность, чел.	0,2	0,32	0,46	0,26	0,2	1,44
Продолжительность участия в разработке ПО (дн.):						
Руководителя	14	20	30	10	8	82
Программиста без категории	10	17	24	20	16	87

Основная заработная плата определяется на основании разряда, тарифной ставки и отработанного времени.

Основная заработная плата исполнителей на конкретное ПО определяется за фактически отработанное время по формуле:

$$ЗП_{\text{осн}} = \sum_{i=1}^n C_{\text{час}_i} \cdot \Phi_{\text{эф}_i} \cdot T_p \cdot K_{\text{пр}}, \quad (6.14)$$

где n — количество исполнителей, занятых разработкой конкретного ПС;

$C_{\text{час}_i}$ — часовая тарифная ставка i -го исполнителя (руб.);

$\Phi_{\text{эф}_i}$ — эффективный фонд рабочего времени i -го исполнителя (дн.);

T_p — количество часов работы в день (час.);

$K_{\text{пр}}$ — коэффициент премирования.

Дополнительная заработная плата включает в себя выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью работников).

Определить дополнительную зарплату можно по формуле:

$$ЗП_{\text{доп}} = Н_{\text{д}} \cdot ЗП_{\text{осн}} , \quad (6.15)$$

где $Н_{\text{д}}$ — норматив дополнительной заработной платы.

Определение часовой тарифной ставки:

$$C_{\text{ч}}^1 = \frac{C_{\text{м}}^1 \cdot 12}{P_{\text{рв}}} = \frac{118 \text{ тыс.} \cdot 12}{2024 \text{ ч}} = 0,699 (\text{тыс. руб.}), \quad (6.16)$$

где $C_{\text{м}}^1$ — месячная тарифная ставка рабочего 1 разряда (118 000 руб.);

$P_{\text{рв}}$ — средняя норма продолжительности рабочего времени за год, определяется по производственному календарю (при 40-часовой рабочей неделе в 2011 г. — 2024 ч.).

Основная зарплата (гр.8, таблица 6.4) вычисляется по формуле:

$$ЗП_{\text{осн}} = C_{\text{ч1}} * K_{\text{т}} * \Phi_{\text{эф}} * 8 * K_{\text{пр}} . \quad (6.17)$$

Дополнительная зарплата (гр. 9, таблица 6.4) вычисляется по формуле:

$$ЗП_{\text{доп}} = Н_{\text{д}} * ЗП_{\text{осн}} . \quad (6.18)$$

Рассчитаем заработную плату разработчикам и результаты занесем в таблицу 6.4.

Таблица 6.4 — Расчет заработной платы

Категории работников	Разряд	Тарифный коэффициент (Кт)	Фэф, дн.	Коэффициент премирования (Кпр)	Нд	Зарплата, тыс.руб.		
						Основная	Дополнительная	Всего
Руководитель	14	3,25	82	1,40	0,15	2 055,479	308,322	2 363,801
Программист без категории	9	2,32	87	1,3	0,15	2 086,375	312,956	2 399,311
ИТОГО						4 141,854	621,278	4 763,132

6.3 Расчет себестоимости и отпускной цены программного обеспечения

Стоимостная оценка ПС у разработчиков предполагает составление сметы затрат, которая включает следующие статьи:

- основная заработная плата исполнителей;

- дополнительная заработная плата исполнителей;
- отчисления в фонд социальной защиты населения;
- налоги, включаемые в себестоимость (исчисляемые от заработной платы);
- материалы;
- спецоборудование;
- машинное время;
- расходы на научные командировки;
- прочие расходы;
- накладные расходы;
- затраты на сопровождение и адаптацию программного средства;
- отчисления в инновационный фонд.

Отчисления в фонд социальной защиты населения ($\Phi_{\text{сзн}}$) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной заработной платы исполнителей. Норматив отчислений составляет 35 %.

Налоги, входящие в себестоимость (исчисляемые от заработной платы), (Н) определяются в соответствии с законодательными актами. В 2011 г. в себестоимость включаются — чрезвычайный налог (3 %), отчисления в фонд занятости (1 %) от основной и дополнительной заработной платы исполнителей.

Расходы по статье «МАТЕРИАЛЫ» (М) определяются на основании сметы затрат, разрабатываемой на программное средство с учетом действующих нормативов.

Норма расхода материалов (H_m) определяется в расчете на 100 машинных команд. При использовании ПС применяется коэффициент снижения среднего расхода материала в пределах от 0.4 до 0.7.

Затраты на материалы (руб.) определяются по формуле:

$$M = H_m \cdot \frac{V_o}{100}, \quad (6.19)$$

Расходы по статье «СПЕЦОБОРУДОВАНИЕ» ($OB_{\text{спец}}$) включают затраты средств на приобретение вспомогательных специального назначения технических и программных средств, необходимых для разработки конкретного программного средства, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию.

Данная статья включается в смету расходов на разработку ПС в том случае, когда приобретаются специальное оборудование или специальные программы, предназначенные для разработки и создания только данного программного средства.

Расходы по статье «МАШИННОЕ ВРЕМЯ» ($BP_{\text{маш}}$) включают оплату машинного времени, необходимого для разработки и отладки ПС.

Машинное время определяется по нормативам в машино-часах на 100 команд машинного времени в зависимости от характера решаемых задач и типа ПЭВМ по формуле:

$$ВР_{\text{маш}} = Ц_{\text{маш-час}} \cdot \frac{V_o}{100} \cdot Н_{\text{мв}} \quad (6.20)$$

где $Ц_{\text{маш-час}}$ - цена машино-часа (см. формулу 6.21);

$Н_{\text{ми}}$ – норматив.

При использовании ПС применяется коэффициент снижения среднего расхода машинного времени в пределах от 0.3 до 0.6.

Цена одного машино-часа работы ПК рассчитывается исходя из часовой заработной платы работника, пользующегося данным компьютером, расходов на электроэнергию и амортизация ПК. Также в цену включается налог на прибыль, единый налог и налог на добавленную стоимость:

$$Ц_{\text{маш. час.}} = \left(C_{\text{час}} * Km^{10} * Knp + 0,5 \text{ кВт} * Ц_{\text{кВт}} + \frac{C_{\text{пк}} * 0,1}{12 * 168} \right) * (1 + Cn) * (1 + Cед) * (1 + Cндс) \quad (6.21)$$

где $Ц_{\text{кВт}}$ — цена за 1 кВт.час., руб.

$C_{\text{пк}}$ — стоимость ПК, руб.

$C_{\text{п}}$ — ставка налога на прибыль,

$C_{\text{ед}}$ — ставка единого налога;

$C_{\text{ндс}}$ — ставка НДС.

- Расходы по статье «НАУЧНЫЕ КОМАНДИРОВКИ» (НК) на конкретное программное средство определяется по нормативу в процентах к основной заработной плате:

$$НК = \frac{H_{\text{нк}}}{100} * ЗП_{\text{осн}}, \quad (6.22)$$

где $H_{\text{нк}}$ - норматив, %.

- Расходы по статье «ПРОЧИЕ ЗАТРАТЫ» (ПрЗ) на конкретное программное средство включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу в процентах к основной заработной плате:

$$ПрЗ = \frac{H_{\text{пз}}}{100} * ЗП_{\text{осн}}, \quad (6.23)$$

где Нпз - норматив прочих затрат, %.

- Затраты по статье «НАКЛАДНЫЕ РАСХОДЫ» (НР) связаны с содержанием аппарата управления, вспомогательных хозяйств и экспериментальных производств, общехозяйственными расходами.

Накладные расходы относятся на конкретное программное средство по нормативу в процентном отношении к основной заработной плате исполнителей:

$$НР = \frac{H_{нр}}{100} * ЗП_{осн}, \quad (6.24)$$

где Н_{нр} - норматив накладных расходов, %.

Сумма 9-ти статей определяет производственную себестоимость ПС:

$$C_{пр} = ЗП_{осн} + ЗП_{доп} + ОФ_{сзн} + Н + М + ОБ_{спец} + ВР_{маш} + НК + ПрЗ + НР \quad (6.25)$$

Кроме того, организация разработчик осуществляет затраты на сопровождение и адаптацию программного средства (З_{са}). Эти затраты определяются по нормативу в процентах от производственной себестоимости:

$$З_{са} = \frac{H_{са}}{100} * C_{пр} \quad (6.26)$$

где Н_{са} — норматив отчислений, %.

Полная себестоимость разработки программного средства определяется по формуле:

$$C_{пол} = C_{пр} + З_{са} + ИФ, \quad (6.27)$$

где ИФ — отчисления в инновационный фонд, определяются по ставке 0.25 % (2011 г.) от суммы производственной себестоимости и затрат на сопровождение и адаптацию:

$$ИФ = 0.0025 * (C_{пр} + З_{са}) \quad (6.28)$$

Выполним расчет себестоимости в табличной форме (см. таблицу 6.5).

Дополнительные данные для расчета:

$$Ц_{маш.час} = (699 \cdot 2,785 \cdot 1,35 + 0,5 \cdot 173 + \frac{2000000 \cdot 0,1}{12 \cdot 168}) \cdot 1,24 \cdot 1,02 \cdot 1,2 \approx 4270 \text{ (руб)} = 4,27 \text{ (тыс. руб.)},$$

					ДП.АС24.06791 – 11 81 00	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

$C_{\text{квт}}$ — 173 руб., $C_{\text{пк}}$ — 2 000 000 руб., ставка налога на прибыль — 24%, ставка единого налога — 2%, ставка НДС — 20%.

Определение отпускной цены. Отпускная цена предприятия - это цена, при которой обеспечивается возмещение текущих затрат производства и получение прибыли.

Таблица 6.5 — Расчет себестоимости ПО

№ п/п	Наименование статей затрат	Норматив	Формула подсчета	Ед.изм.	Сумма
A	B	C	D	E	F
1	Основная заработная плата разработчиков	—	—	тыс.руб.	4 141,854
2	Дополнительная заработная плата	—	—	тыс.руб.	621,278
3	Отчисления в фонд социальной защиты населения	35%	$F3 = 0.35 \cdot (F1 + F2)$	тыс.руб.	1667,096
4	Налоги, исчисляемые от заработной платы, в т.ч.	—	$F4 = F4.1 + F4.2$	тыс.руб.	190,525
4.1	Чрезвычайный	3%	$F5 = 0.03 \cdot (F1 + F2)$	тыс.руб.	142,894
4.2	Фонд занятости	1%	$F6 = 0.01 \cdot (F1 + F2)$	тыс.руб.	47,631
5	МАТЕРИАЛЫ (формула 6.19)	—	$F5 = 0,699 \cdot 4260 : 100$	тыс.руб.	29,778
6	СПЕЦОБОРУДОВАНИЕ	—	—	—	не применяется
7	МАШИННОЕ ВРЕМЯ (формула 6.20)	—	$F7 = 4,26 \cdot 4260 : 100 \cdot 0,6$	тыс.руб.	108,886
8	НАУЧНЫЕ КОМАНДИРОВКИ	—	—	—	не применяется
9	ПРОЧИЕ ЗАТРАТЫ (формула 6.23)	20%	$F9 = 0.2 \cdot F1$	тыс.руб.	828,371
10	НАКЛАДНЫЕ РАСХОДЫ (формула 6.24)	40%	$F10 = 0.4 \cdot F1$	тыс.руб.	1656,742

Продолжение таблицы 6.5 – Расчет себестоимости ПО

A	B	C	D	E	F
11	Производственная себестоимость	—	$F11 = F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8 + F9 + F10$	тыс.руб.	9244,53
12	Расходы на сопровождение и адаптацию	15%	$F12 = 0.15 \cdot F11$	тыс.руб.	1386,68
13	Инновационный фонд	—	$F13 = 0.0025 \cdot (F11 + F12)$	тыс.руб.	26,578
14	Полная себестоимость	—	$F14 = F11 + F12 + F13$	тыс.руб.	10657,788

Организация-разработчик не финансируется из бюджета и не имеет льгот в соответствии с Указом Президента № 324.

Выполним расчет отпускной цены в табличной форме (см. таблицу 6.6).

Таблица 6.6 — Расчет отпускной цены

№ п/п	Наименование статей затрат	Норматив	Расчетная формула	Сумма затрат, тыс.руб.
A	B	C	D	E
1	Полная себестоимость	—	—	10657,788
2	Прибыль	20%	$\frac{1E * 2C}{100}$	2131,558
3	Прогнозируемая цена без налогов	—	$1E + 2E$	12789,346
4	Республиканский единый платеж	2%	$\frac{3E * 4C}{100\% - 4C}$	261,007
5	НДС	20%	$\frac{3E * 5C}{100\%}$	2557,869
6	Отпускная цена	—	$3E + 4E + 5E$	15608,222

Прибыль от реализации программного средства за вычетом налога на прибыль остается организации-разработчику и представляет собой эффект от создания нового программного средства вычислительной техники, вычисляется по формуле (6.29):

$$П_{\text{ч}} = П_{\text{р}} - Н_{\text{пр}}. \quad (6.29)$$

где $Н_{\text{пр}}$ — налог на прибыль, рассчитывается по ставке 24% и вычисляется по формуле (6.30):

$$H_{np} = 0,24 \cdot \Pi_p. \quad (6.30)$$

Подставляя исходные данные в формулы (6.29) и (6.30), получим:

- 1) налог на прибыль равен $H_{np} = 0,24 \cdot 2131,558 = 511,581$ (тыс. руб.);
- 2) чистая прибыль равна $\Pi_q = 2131,588 - 511,581 = 1620,007$ (тыс. руб.).

В результате полученных вычислений можно сделать вывод, что разработанное ПС экономически целесообразно создавать, так как помимо возможности навигации мобильного робота на основе данной системы, также получим прибыль.

					ДП.АС24.06791 – 11 81 00	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

7 ЭНЕРГО И РЕСУРСОСБЕРЕЖЕНИЕ

В настоящее время используемые человечеством энергоресурсы постепенно иссякают, стоимость их добычи увеличивается, а нерациональное использование сказывается на экологии.

Энергосбережение (экономия электроэнергии) — реализация правовых, организационных, научных, производственных, технических и экономических мер, направленных на эффективное (рациональное) использование (и экономное расходование) топливно-энергетических ресурсов и на вовлечение в хозяйственный оборот возобновляемых источников энергии. Энергосбережение — важная задача по сохранению природных ресурсов.

Проблема снижения энергетических затрат, проблема энергосбережения становится все более актуальной в мировом аспекте. Эта проблема еще более обостряется в связи с постоянным увеличением стоимости энергоносителей: природного газа, нефтепродуктов, электроэнергии и т.д.

При решении проблем энергосбережения важно определить основные стратегические подходы и методы рационального использования энергоресурсов, которые могут быть как общими для всей экономики, так и специфичными для отдельных отраслей промышленности, сельского хозяйства и социальной сферы. Среди таких наиболее общих подходов в стратегии энергосбережения можно было бы назвать применение ресурсосберегающих технологий в сфере энерготехнологических объектов, использование методов математического моделирования и оптимизации при проектировании и реконструкции предприятий различных отраслей промышленности, замену дорогостоящих энергоемких видов энергоносителей, таких как электроэнергия, кокс на более дешевые, в частности, на природный газ, все более широкое использование возобновляемых источников энергии - ветра, солнца, биомассы и др.

Несмотря на имеющуюся литературу по проблемам энергосбережения, в том числе и выпускаемые периодические журналы, все же освещение этих вопросов остается недостаточным. Это затрудняет как принятие обоснованных решений в области энергосбережения, так и обеспечение соответствующего кадрового сопровождения.

Необходимо изучить блок вопросов, связанных с законодательно-правовой базой в энергосбережении, стандартами, лицензированием, паспортизацией, энергоаудитом, нормированием и тарифообразованием на энергоносители.

В настоящее время наиболее насущным является бытовое энергосбережение (энергосбережение в быту), а также энергосбережение в сфере жилищно-коммунального хозяйства. Препятствием к его осуществлению является сдерживание роста тарифов для населения на отдельные виды ресурсов (электроэнергия, газ), отсутствие средств у предприятий жилищно-коммунального хозяйства на реализацию энергосберегающих

программ, низкая доля расчетов по индивидуальным приборам учета и применение нормативов, а также отсутствие массовой бытовой культуры энергосбережения.

Актуальным также является обеспечение энергосбережения в агропромышленном комплексе.

В 2008 году число используемых во всем мире ПК превысило 1 миллиард, и, по прогнозам специалистов, к 2014 году это число удвоится. Как известно, многие миллиарды долларов в год пропадают впустую из-за потерь электроэнергии за счет неэффективных источников питания личных и домашних электронных приборов, многие из которых продолжают использовать энергию даже тогда, когда они выключены, но не отсоединены от сети: «вокруг нас находятся настоящие электронные вампиры, которые высасывают электричество день и ночь». Однако современные технологии не стоят на месте и многие компьютерные фирмы и корпорации внедряют энергосберегающие технологии. Приведём несколько примеров

Фирма Foxconn задействовала три направления в решении энергосбережения, объединенные общим названием 3G (три «зеленые» технологии): GoD (Green on Demand), GPS (Green Power Saving) и GSM (Green System Mode). Технология Green on Demand обеспечивает экономию 20,5% электроэнергии по сравнению со стандартными системами за счет отключения ненужных в данный момент линий питания процессора в периоды его простоя или сниженной нагрузки. Технология Green Power Saving снижает энергопотребление в дежурном режиме до 99,4% (с 8,1 до 0,05 Вт). GSM является инструментом автоматизированной настройки системы на сниженное энергопотребление. Foxconn иллюстрирует эффективность своих технологий следующим примером: ПК с поддержкой технологий энергосбережения 3G, находящийся 20 часов в дежурном режиме, два часа работающий под высокой нагрузкой и два часа простаивающий, экономит 85% энергии. По данным Foxconn, в среднем компьютеры находятся 15 часов в дежурном режиме и шесть часов в режиме простоя. Если бы все эти компьютеры были оснащены средствами энергосбережения Foxconn, то мировая экономия электроэнергии составила бы около 130 млрд кВт/ч.

Энергосберегающие функции сетевого фильтра GreenPower MDP 900 от Monster Cable позволяют экономить существенное количество энергии, когда подключенные приборы уходят в режим standby. Наиболее полезно это устройство в сочетании с настольным компьютером. Когда компьютер переключается в спящий режим, MDP 900 автоматически отключает от сети все периферийные устройства, такие как принтер или монитор, а когда компьютер выходит из спящего режима, подключает их обратно. По мнению создателей, устройство может снизить затраты на электроэнергию (США) на \$130 в год, то есть на цену самого сетевого фильтра.

Мощная современная видеокарта под полной нагрузкой требует столько же энергии, сколько остальные комплектующие ПК вместе взятые: от 110 до 270 Вт. Поэтому производители приступили к выпуску интеллектуальных видеокарт с управлением потребления электроэнергии в зависимости от нагрузки.

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

Многие годы человек проводит на работе или в учебном заведении до трети времени в рабочие дни. Конечно, на обеспечение комфортных условий работы в офисе и выполнение различных рабочих операций требуется не так много энергии, как на выплавку стали или нефтепереработку.

Тем не менее, и в офисе есть много потребляющего энергию оборудования и большие возможности для энергосбережения.

Причем в отличие от выплавки стали, использовать эти возможности достаточно просто и зачастую это не требует больших затрат или не стоит ничего и почти всегда в конечном итоге оборачивается выгодой для сотрудников и компании.

А эффект от большинства простых и доступных мер по энергосбережению в офисе будет получен практически немедленно.

Известно, что через плохо утепленные окна может теряться до половины тепла.

Чтобы привести окна в порядок, не обязательно устанавливать дорогостоящие стеклопакеты. В большинстве случаев достаточно утеплить их современными изоляционными материалами.

Даже в самые лютые морозы вполне достаточно, чтобы температура в помещении держалась на уровне 18-20 градусов.

По возможности установите регуляторы на радиаторы отопления и регулируйте температуру в зависимости от погоды. Следите за температурой в помещении и на улице с помощью термометра. В некоторых помещениях, например, в спальне, коридорах, редко используемых комнатах, температура может быть даже ниже.

Если в офисе топят слишком сильно, необходимо обратиться в ЖЭС или другую организацию, оказывающую жилищно-коммунальные услуги, с просьбой установить специальную аппаратуру, регулирующую подачу тепла в дом. Снижение температуры в доме на 1 градус позволяет сократить сжигание топлива и снизить выбросы парниковых газов на 300 килограммов.

Необходимо использовать для освещения помещений энергосберегающие люминесцентные лампы. Сегодня они стоят уже не так дорого и по карману большинству покупателей, хотя и дороже обычных ламп накаливания. Но дополнительные затраты на их покупку окупятся за короткое время за счет экономии на платежах за электричество и длительного срока службы. Использование таких ламп значительно снижает энергопотребление.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проектирования была разработана система детектирования мобильного робота и препятствий для его движения, интегрируемая в систему управления мобильным роботом.

Были пройдены все этапы разработки программного обеспечения от анализа задачи, до тестирования, показавшего работоспособность системы.

При создании были решены следующие поставленные задачи:

- 1) детектирование положения и ориентации мобильного робота;
- 2) определение координат мобильного робота;
- 3) расчет пройденного расстояния;
- 4) генерации траектории пройденного пути робота.

Система разработана с учетом удобства последующего сопровождения и модификации. Это обеспечивается использованием объектно-ориентированного подхода программирования, позволяющего модифицировать отдельные классы алгоритмы, без модификации всей системы в целом, а так же модульной архитектурой системы.

Для удобства тестирования и наладки системы, реализована возможность ручного управления испытанием и задание режима автономного испытания.

Так же в завершении выполнения дипломного проекта был рассчитан экономический эффект от разработки. В результате расчета, чистая прибыль составила 1 620,007 тыс. руб. Отпускная цена программного продукта составила 15 608,222 тыс. руб., а его себестоимость — 10 657,788.

Вывод по расчету экономических показателей: в результате проведения расчета экономического эффекта от внедрения разработки было установлено, что в процессе использования чистая прибыль в конечном итоге возмещает капитальные затраты.

					<i>ДП.АС24.06791 – 11 81 00</i>	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

СПИСОК СОКРАЩЕНИЙ

АО — аппаратным обеспечением
БИНС — бесплатформенная инерциальная навигационная система
ИК — инфракрасный
ИНС — инерциальная навигационная система
ОС — операционную систему
ПО — программное обеспечение
ПЭВМ — персональной электронной вычислительной машины
ASCII — American Standard Code for Information Interchange
BSD — Berkeley Software Distribution
DOM — Document Object Model
FTP — File Transfer Protocol
GNU — GNU's Not UNIX
GPL — General Public License
GPS — Global Positioning System
GSM — Global System for Mobile Communications
HTTP — HyperText Transfer Protocol
LGPL — Lesser General Public License
MOC — Meta Object Compiler
OpenCV — Open Source Computer Vision Library
QR-код — Quick Response код
RFID — Radio Frequency Identification
SAX — Simple API for XML
SDK — Software Development Kit
SLAM — Simultaneous localization and mapping
SLAM — Simultaneous Localization and Mapping
SQL — Structured Query Language
SVG — Scalable Vector Graphics
TCP/IP — Transmission Control Protocol / Internet Protocol
UART — Universal asynchronous receiver/transmitter
UDP — User Datagram Protocol
Wi-Fi — Wireless Fidelity
WYSIWYG — What You See Is What You Get
XML — eXtensible Markup Language

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дёмин В.В., Дунец А.П., Модель агента игрока робофутбольной команды - Брест: БрГТУ, 2010. - с. 242-245
2. Borenstein J., Everett H. R., Where am I? Sensors and Methods for Mobile Robot Positioning - Michigan: University of Michigan, 1996. - 282с.
3. Ernest P., Mázl R., Preucil L., Train Locator Using Inertial Sensors and Odometer - University of Parma, Italy: IEEE Intelligent Vehicles Symposium, 2004. - с. 860-865
4. Seo J., Lee H.K., Lee, J.G and Park C.G., Lever Arm Compensation for GPS/INS/Odometer Integrated System - Automat. and Systems: Inter. Journal of Contr, 2006. - с. 247-254
5. Thompson W.B., Henderson T.C., Vision-Based Localization - Proc. Image Understanding Workshop 93: , 1993. - с. 491-498
6. Durrant-Whyte H., Bailey T., Simultaneous Localization and Mapping: Part I - IEEE Robotics & Automation Magazin: IEEE, 2006. - с. 99-108
7. Дёмин В.В., Дунец А.П., Реализация модели робота игрока для Robocup Soccer Simulation Server - Брест: БрГТУ, 2010. - с. 255-259
8. Шлее М., Qt4. Профессиональное программирование на С++ - Санкт-Петербург: БХВ-Петербург, 2007. - 880с.
9. Бланшет, Ж., QT4: программирование GUI на С++ - Москва: КУДИЦ-ПРЕСС, 2007. - 628с.
10. Bradski, G., Kaebler, A., Learning OpenCV. Computer Vision with the OpenCV Library - Gravenstein Highway North, Sebastopol: O'Reilly, 2008. - 577с.
11. ЕСПД. ГОСТ 19.504 – 78. Руководство программиста. Требования к содержанию и оформлению.
12. ЕСКД. ГОСТ 2.105 – 95. Общие требования к текстовым документам.
13. ГОСТ 34.602 – 89. Техническое задание на создание автоматизированной системы.
14. Брудник С.С., Кочегарова И.А., Степин Ю.П., Определение экономической эффективности программных средств в АСУ - М.: ГАНГ, 1995.

ПРИЛОЖЕНИЕ А — ТЕКСТ ПРОГРАММЫ